Searched the web for **implementation "total quality management" "continuous improvement""capability maturity"**. Results

Verizon Information Technologies | **Capability Maturity** Model
... you in achieving **total quality management** best practices ... CMM consulting and **implementation**
services, contact ... be committed to **continuous improvement** changes and ...
www.verizonit.com/aboutus/cmm.htm - 34k - Mar 3, 2003 - Cached - Similar pages

Q/P Management Group, Inc. Solutions for Improving Quality and ...
... and Measurement Techniques **Total Quality Management** Planning and ... Training and **Implementation** The SEI ... **Maturity** Model **Continuous Improvement** in Software ...
www.qpmg.com/sitemap.htm - 23k - Cached - Similar pages

[PDF]Quality Journey
File Format: PDF/Adobe Acrobat - View as HTML
... ITSD QMS ISO 9001 **Total Quality Management** (TQM) ... matters Quality Committee **Continuous**
Improvement Management Leadership & ... Support ITBB in **Implementation** of the ...
www.itsd.gov.hk/itsd/english/pubpress/ download/esp020315.pdf - Similar pages

Define and describe the **capability maturity** model (CMM) as it is ...
... of performance and **continuous improvement** in their ... [4] **Capability Maturity** Model® (SW ... Relationships Between **Total Quality Management Implementation** Factors And ...
cas.uah.edu/templeton/CapabilityMaturityModel.htm - 16k - Cached - Similar pages

Addison-Wesley
... and predictable using **Total Quality Management** concepts pioneered ... include a roadmap for **implementation**. ... of professionalism dedicated to **continuous improvement**. ...
www.awprofessional.com/catalog/ article.asp?product_id=%7B697CE970-4A6C-4924-B142-ED40C8991A63%7D - 17k - Cached - Similar pages

Quality Process - TBI Ltd
... **Total Quality Management** principles applied to the Information ... we look at the **implementation**
of a ... and provides feedback, and drives **continuous improvement**. ...
www.targetimprovement.co.uk/titjqpweb.html - 77k - Cached - Similar pages

Results
... 6, Facilitating **implementation** of **total quality management** through information ... 10, Toward a theory of **continuous improvement** and the learning curve Willard I ...
portal.acm.org/ results.cfm?query=%22total%20quality%20management%22%20%3CIN%3E%20keyword&coll=GUIDE&a... - Similar pages

Welcome to Improve QPI - affordable tools, services and training ...
... act as the starting point for **continuous improvement** and/or **Total Quality Management** initiatives. ... is capable of performing most of the **implementation** tasks; ...
www.improveqpi.co.uk/services/qmsimpl.htm - 14k - Cached - Similar pages

[PPT]www.systemthinker.com/Presentations/MPTQMSalesPresentation.ppt

File Format: Microsoft Powerpoint 97 - View as HTML
... The Mail Preparation **Total Quality Management** (MPTQM) program is ... 10. System **Implementation** Model. ... of quality models, **continuous improvement**, management commitment ...
Similar pages

[PDF]Mapping Knowledge Reuse in R&D Processes - The **Implementation** of ...
File Format: PDF/Adobe Acrobat - View as HTML
... **Total Quality Management** (TQM) & Total Reuse Management (TRM ... the concept of **continuous improvement** [Juran ... solution and standardize **implementation** Itirate figure 4 ...
www.cranfield.ac.uk/sims/ecotech/pdfdoc/ reuse_quality_conf98.pdf.prn.pdf - Similar pages

Goooooooooogle ▶
Result Page: **1** 2 3 4 5 6 7 8 9 10 **Next**

Searched the web for **ipd-cmm**. Results **21** - **30** of about **636**. Search took **0.16** seconds.

[PPT]SE-CMM Provides Process Capability Framework for Engineering ...
File Format: Microsoft Powerpoint 97 - View as HTML
... ISO. 15504. (SPICE). People. CMM. **IPD-**. **CMM***. DOD. IPPD. SECAM. AF IPD. Guide. ... 1996.
SECAM. 1996. SW-CMM V2.0 Draft C*. 1997. **IPD-CMM** V0.98*. 1997. * Not released. CMMI.
EIA/IS 731. ...
www.sdincose.org/html/Presentations/ Wuersch_SE_CMM/WuerschSECMM1111.ppt - Similar pages

[PDF]Microsoft PowerPoint - CMMI R2 INCOSE TSK Staged Vs. Continuous
File Format: PDF/Adobe Acrobat
... Page 6. 6 Why Do We Have Two Representations? * Source Model Heritage Software
CMM--Staged SECM--Continuous **IPD CMM**-Hybrid * Proponents for each type ...
www.sdincose.org/html/Presentations/Tom%20Kudlick/ CMMI%20R2%20INCOSE%20TSK%20Staged%20Vs.pdf - Similar pages
[More results from www.sdincose.org]

[PPT]This briefing is UNCLASSIFIED
File Format: Microsoft Powerpoint 97 - View as HTML
... SW-CMM V2.0C (draft). SECM. **IPD-CMM**. Multiple CMMI models planned. ... (3) Integrated
Product Dev. Capability Maturity Model, draft V0.98 (**IPD-CMM**). GENERAL DYNAMICS. ...
www.gd-ais.com/sv-spin/Events/Presentations/ GD-CMMI_Project.ppt - Similar pages

STSC CrossTalk - Creating an Integrated CMM for Systems and ...
... SECM) Electronic Industries Association Interim Standard 731 (EIA/IS 731), and the Integrated Product Development Capability Maturity Model (**IPD-CMM** v0.98). ...
www.stsc.hill.af.mil/crosstalk/2000/09/phillips.html - 24k - Cached - Similar pages
[More results from www.stsc.hill.af.mil]

CMM
... SA-CMM (Software Acquisition CMM) , 人に関するp-CMM (people-CMM) , 製品開発に関する**IPD-CMM** (Integrated Product Development CMM ...
www.itscj.ipsj.or.jp/tutorials/tu53.html - 17k - Cached - Similar pages

CMM-2
... TCM,PCM は , SW- CMM 1.1版のKPAの略称 . CF.はCommon Featureの略 . - は新規 . 注2: ()内の記号説明: SEはSE-CMM , IPDは**IPD-CMM** , SW-CMMv2C ...
www.sra.co.jp/public/doc/GSletter/ vol.30/CMM/CMM-6.html - 3k - Cached - Similar pages
[More results from www.sra.co.jp]

[PPT]www.cs.ccu.edu.tw:8080/speech/CMM.ppt
File Format: Microsoft Powerpoint 97 - View as HTML
... Hybrid. **IPD-CMM**. System engineering. Continuous. Sys. ... SW-CMM V2 draft C. EIA Interim Standard 731, System Engineering capability Model. **IPD-CMM** draft v0.98. 1/4/03. ...
Similar pages

Capability Maturity Model
... Integrated Product Development CMM (**IPD-CMM**). http://www.sei.cmu.edu/cmm/**ipd-cmm**.html. CMM Integration CMMI. http://www.sei.cmu.edu/cmmi/. ...
www.iturls.com/English/SoftwareEngineering/SE_13.asp - 101k - Cached - Similar pages

[PDF]Overview of the CMMI
File Format: PDF/Adobe Acrobat - View as HTML
... Different representations ! Multiple assessments ! Multiple training ! Multiple expenses P-CMM SW-CMM Security-CMM **IPD-CMM** SE-CMM SA-CMM Page 6. ...
www.ifi.uio.no/in331/foiler/h01/cmmipres.pdf - Similar pages

Links1
... Other. Process Models. SEI SW - CMM. SEI SE - CMM. SEI CMMI. SEI - **IPD CMM**. P - CMM. SA - CMM. Personal Software Process. Project Management Maturity Model. SPINS. ...
www.ieee-cs-cts.org/links.htm - 34k - Cached - Similar pages

◀ Goooooooooooogle ▶
Result Page: **Previous** 1 2 **3** 4 5 6 7 8 9 10 11 12 **Next**

Searched the web for **product improvement project management capability model**. Results **1 - 10** of about **219,000**. Search tc

[PDF]Introduction: **Capability** Maturity **Model** Integration
File Format: PDF/Adobe Acrobat - View as HTML
... and approaches to process **improvement** is necessary. ... SECM)] and integrated **product**
development (IPD ... deal with requirements, **project management**, process definition ...
interactive.sei.cmu.edu/Features/1998/September/ Introduction/introduction_sept98.pdf - Similar pages

Capability Maturity **Model** Integration
... both deal with requirements, **project management**, process definition ... one CMM to improve
product quality and ... apply both models in integrated process **improvement**. ...
interactive.sei.cmu.edu/Features/1998/september/ Introduction/introduction_sept98.htm - 22k - Cached - Similar pages
[More results from interactive.sei.cmu.edu]

STSC Consulting Services - CMMI - Documentation - **Capability** ...
... Quantitative **Project Management** (QPM), Collects **project** process and **product** metrics,
and analyzes results to identify process **improvement** opportunities. ...
www.stsc.hill.af.mil/consulting/cmmi/more_cmmi.html - 22k - Cached - Similar pages

Software **Capability** Maturity **Model**
... the software process and **product** quality are ... Continuous process **improvement** is enabled
by quantitative ... related to establishing basic **project management** controls ...
www.sei.cmu.edu/cmm/cmm.sum.html - 18k - Mar 3, 2003 - Cached - Similar pages

CMMI Version 1.1 Tutorial
... The CMMI **Project** CMMI Models Comparing **Model** ... of the Continuous Representation CMMI
Product Suite CMMI ... Line 2 Categories of Process **Improvement** Benefits Results ...
www.sei.cmu.edu/cmmi/presentations/euro-sepg-tutorial/ - 27k - Mar 3, 2003 - Cached - Similar pages
[More results from www.sei.cmu.edu]

[PDF]Software Process **Improvement** Using The **Capability** Maturity **Model** ...
File Format: PDF/Adobe Acrobat - View as HTML
... Establish goals for process **improvement** (To Be ... Process Continuously Improving Process
Project Management Process Architecture **Product** Line **Management** ...
www.sasqag.org/pastmeetings/SASQAGvu.PDF - Similar pages

Process **Improvement**: The **Capability** Maturity **Model** - itmWEB ...

... level, at least basic **project management** activities are ... Seeking formal assessment and **improvement** under this ... quality of the software **product** and predictability ...
www.itmweb.com/f051098.htm - 10k - Cached - Similar pages

Continuous Staged Process **Management** Organizational Process Focus ...
... Appraisal Method for Process **Improvement** (SCAMPI), Version ... Integrated **Product** and Process Development (IPPD) Tutorial; ... The CMM Integration **Project** presentation ...
seir.sei.cmu.edu/seir/frames/body2.map.CMMI.html - 21k - Cached - Similar pages

Software Process **Improvement** (SPI):SEI Process **Improvement** Models
... products to support process and **product improvement**. ... of CMM, Process **Improvement**, **project management**, and software ... **Capability** Maturity **Model** for Software (SW ...
www.dacs.dtic.mil/databases/url/ key.hts?keycode=39:2582&islowerlevel=1 - 16k - Mar 3, 2003 - Cached - Similar pages

IT **Project** **Management**, Software Engineering, Personal Software ...
... They cover IT **project management**, the delivery of IT ... IPD-CMM Integrated **Product** Development **Capability** ... Software Process **Improvement**: Practical Guidelines for ...
www.technosphere.net/profdev.htm - 16k - Mar 3, 2003 - Cached - Similar pages

Goooooooooogle ▶
Result Page: **1** 2 3 4 5 6 7 8 9 10 **Next**

product improvement project manag | Google Search | Search within results

Google Advanced Search Preferences Language Tools Search Tips

accenture accelerating implementing software CMM

Google Search

Web Images Groups Directory News

Searched the web for **accenture accelerating implementing software CMM**. Results 1 - 10 of about 25. Search

Cmm - Company-Wide Strategic Planning & Project Management. Sponsored Link
www.systemcorp.com Get Enterprise Project and Portfolio Management - Get Info!

Cmms **Software**. Sponsored Link
www.managestar.com Web-based Cmms **software** Fortune 500 clients

QAI India Ltd. - Consulting

... and IT organizations for **implementing** SW-**CMM** ... Datamatics, Price Waterhouse, Intel,
Accenture, SISL, Satyam, HP ... around the world, **accelerating** the implementation ...
www.qaiindia.com/Abt_us/abt_us.htm - 20k - Cached - Similar pages

[PDF]in Asia Pac

File Format: PDF/Adobe Acrobat
... the Training Gap' (102HS) > '**Accelerating** RUP Implementation ... an industry consortium
including Satyam & **Accenture**. ... the experiences in **implementing** the model ...
www.qaiindia.com/SEPG_minisite/chin_broch.pdf - Similar pages

Change Management Consultants

... **Accenture** (formerly Anderson Consulting) Consultants in change ... assessment and consulting
for **software** and systems ... to measure your progress in **implementing** TQ. ...
software.isixsigma.com/co/change_management/ - 47k - Cached - Similar pages

[PDF]Management Update: The Latest Trends in Application Outsourcing ...

File Format: PDF/Adobe Acrobat - View as HTML
... Similarly, although **Accenture** has many ... can provide an effec- tive means of **accelerating**
the adoption ... for the work, US vendors are **implementing** global delivery ...
www4.gartner.com/1_researchanalysis/30may2001g.pdf - Similar pages

[PDF]WG IT D S IT SA T

File Format: PDF/Adobe Acrobat
... to the principles of **CMM** and used ... Information systems - **Software** products - **Software**
services - Internet ... Nevertheless the present **accelerating** production and ...
www.e-energia.org/pdf/ WG_ITD_State-of-the-art_Final.pdf - Similar pages

The New Wave

... is a major element of **Accenture**'s strategic ... processing and validation systems,

Sponsored Links

CMM Certification
Achieve a Level 2-5 using SEI registered Lead Assessors
www.bogiso.com
Interest: ~~~~~~~

CMM Measurements
Accurate CMM inspection and parts using Zeiss with Vast Scanning Head
www.fastraknet.com
Interest: ~~~~~~~

How to Implement the **CMM**
Want fast, effective results? Order our book or consulting today!
www.businessprocess-solutions.com
Interest: ~~~~~~~

CMM Compliance Guaranteed
processMax, the industry-leading **software** project management system
www.pragmasystems.com
Interest: ~~~~~~~

Web-based **CMM Software**
Integrates with MS Project & PMI. Source Code, Free Version & Demo
www.vcsonline.com/Enterprise_VPMi
Interest: ~~~~~~~

Implement **Software** Papers
Get Implement **Software** White Papers Download Free Information Here
www.bitpipe.com
Interest: ~~~~~~~

Full **CMM** Services
Practical, proven guidance for projects. Online CMM Ref. Tool
www.TeraQuest.com
Interest: ~~~~~~~

Careers at **Accenture**
Fast-track your career with our consulting advice & insights

Google Advanced Search Preferences Language Tools Search Tips

accelerating implementing software CMM

Google Search

Web Images Groups Directory News

Searched the web for accelerating implementing software CMM. Results 1 - 10 of about 922. Search took 0.06 s

[PDF]CMM ?????
File Format: PDF/Adobe Acrobat
... 2 • **Software** Quality: Some Facts and Figures ... **CMM** • Benefits of **Implementing CMM**
– Some Real ... Principles for **Accelerating CMM** Implementation • Summary ...
www.qaiindia.com/Misc/thought_leadership/cmm_dan.pdf - Similar pages

[PDF]ACCELERATING CHANGE IN SPI FACTSHEET 1 March 1994 Workshops for ...
File Format: PDF/Adobe Acrobat - View as HTML
... a consistent methodology for **implementing** and managing ... changes related to **software**
process improvement. The **Accelerating** Change (AC) methodology developed by ...
www.defenselink.mil/c3i/bpr/bprcd/vol2/0093.pdf - Similar pages

[PDF]**Accelerating** CMMI? Adoption with Technology Adoption Tools
File Format: PDF/Adobe Acrobat - View as HTML
... interactive.sei.cmu.edu 1 **Accelerating** CMMI ® Adoption ... the Capability Maturity Model
for **Software** (SW–**CMM** ... the same transition strategies in **implementing** CMMI ...
interactive.sei.cmu.edu/news@sei/features/2002/3q02/pdf/feature-1-3q02.pdf - Similar pages

news@sei interactive | Features
... **Accelerating** CMMI Implementation with Technology Adoption Tools ... Capability Maturity
Model for **Software** (SW-**CMM** ... same transition strategies in **implementing** CMMI. ...
interactive.sei.cmu.edu/news@sei/features/2002/3q02/feature-1-3q02.htm - 41k - Cached - Similar pages
[More results from interactive.sei.cmu.edu]

Detailed Training and Services R 2.5
... teams, and their performance **implementing** and maintaining ... Services to Support **Accelerating**
Change, Working ... **Software** management process capability assessment as ...
www.patech.com/detsrvcs.htm - 16k - Cached - Similar pages

Google Advanced Search Preferences Language Tools Search Tips

sei ideal model

Google Search

Web Images Groups Directory News

Searched the web for sei ideal model. Results 1 - 10 of about 21,600. Search took 0.16 seconds.

The IDEAL Model

... **IDEAL** Contact Information. Contact us to learn more about the **IDEAL model**, **ideal**-team@**sei**.cmu.edu. Return to top | Return to Management Practices. ...
www.sei.cmu.edu/ideal/ - 14k - Cached - Similar pages

The IDEAL(SM) Model: A Practical Guide for Improvement

... Recognizing that the **model** had great potential outside of the process
arena, the **SEI** has revised the **IDEAL Model** for broader application. ...
www.sei.cmu.edu/ideal/ideal.bridge.html - 26k - Cached - Similar pages
[More results from www.sei.cmu.edu]

Process Inc: SEI CMM/CMMI and ISO 15504 SPICE Software Process ...

... Management Consulting Process Inc provides a series of consulting services based on the **SEI's IDEAL model** of the software process improvement cycle. ...
www.processinc.com/ - 13k - Cached - Similar pages

Capability Maturity Model

... The **IDEAL Model**. http://www.**sei**.cmu.edu/**ideal**/**ideal**.html. The **IDEAL Model**: A Practical Guide for Improvement. http://www.**sei**.cmu.edu/**ideal**/**ideal**.bridge.html. ...
www.iturls.com/English/SoftwareEngineering/SE_13.asp - 101k - Cached - Similar pages

STSC Consulting Services - Tech Adoption Svcs - Documentation - ...

... This **model** was developed by **SEI** to help organizations improve their software processes. At the STSC, we have adopted the **IDEAL Model** as our process improvement ...
www.stsc.hill.af.mil/consulting/ tech_adoption/ideal_model.html - 17k - Cached - Similar pages

Project Suite: Models and Views

... **Ideal Model** Adaptation, **SEI Ideal Model** for Software Process Improvement, Initiating, diagnosing, establishing, acting, leveraging. ...
members.aol.com/ONeillDon/models-views.html - 8k - Cached - Similar pages

An Enhanced Framework for the Management of Information ...

... It should be noted that, to keep the document to a manageable size, details of the **SEI IDEAL** SM **Model** have not been repeated. A ...
www.cio-dpi.gc.ca/emf-cag/ppw-sip/ppw-sip10_e.asp - 23k - Sep 9, 2003 - Cached - Similar pages

Process Model Reference

... P-CAM® is an Appraisal approach designed to support the overall theory and intent expressed by the **SEI's IDEAL** and the P-CAM® process models. ...
www.pep-inc.com/P-CAM%20Method/ideal.htm - 11k - Cached - Similar pages

Date: 2/21/99

... Effort; **SEI**; **Ideal** Improvement **Model**; Aspects of the **Ideal Model**; A Practical Approach; Project Management (PM) - What Is It? Project ...
www.softdim.com/psqt99north/tracks2/ 49%20-%20Streum.html - 7k - Cached - Similar pages

```
Set     Items   Descripti
S1          2   IMPLEMENT?()SOFTWARE()(CMM OR CAPABILITY()MATURITY()MODEL)
S2       8011   ACCENTURE
S3          4   S2 AND CMM
S4          4   RD (unique items)
S5       2275   CMM OR CAPABILITY()MATURITY()MODEL
S6        413   S5 AND PROCESS
S7          9   S6 AND ACCELERATE
S8          8   RD (unique items)
S9          8   S8 NOT PD>011207
?
```

- IDEAL Main Page
- Case Study
- Conceptual Reconstruction
- IDEAL Graphics
- Presentation

The IDEAL[SM] Model: A Practical Guide for Improvement

by Jennifer Gremba and Chuck Myers

This article appeared in the Software Engineering Institute (SEI) publication, *Bridge*, issue three, 1997.

Organizations are increasingly recognizing the need for specific implementation guidance when they adopt new software engineering tools, processes, and methods. Many improvement efforts, including software process improvement, continuous risk management, or the introduction of a new development environment, are so complex, and their effects so far reaching, that they require a specialized, systematic approach for managing the technology adoption life cycle. The SEI has developed and refined the IDEAL model to help satisfy this need.

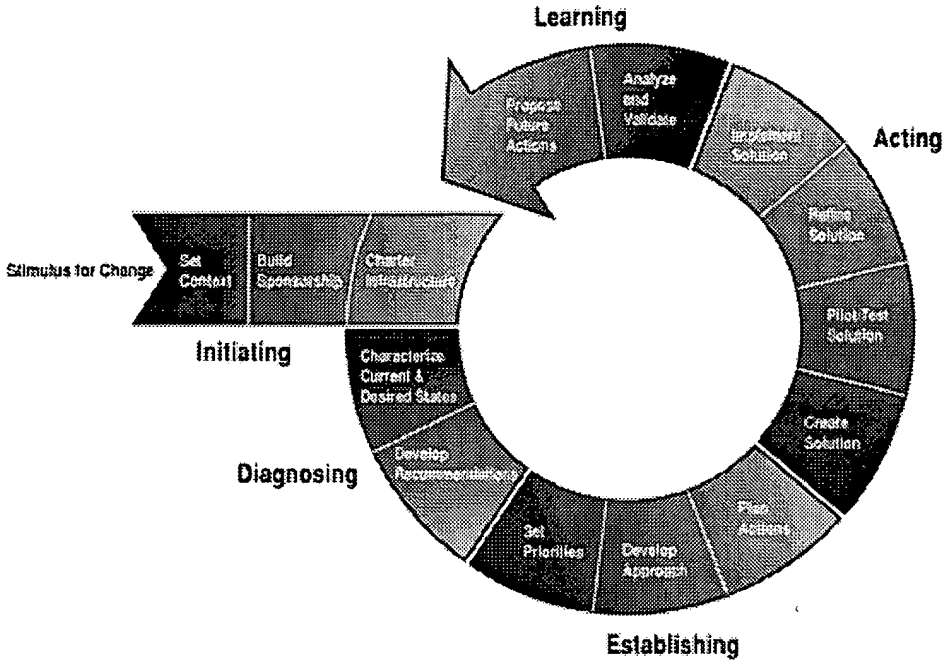
The IDEAL model as originally conceived was a life-cycle model for software process improvement based upon the Capability Maturity Model® (CMM®) for Software, and for this reason the model used process improvement terms. Recognizing that the model had great potential outside of the process arena, the SEI has revised the IDEAL Model for broader application. The new version of the model has been designated as Version 1.1 to emphasize that this is a first step toward the goal of broader applicability.

IDEAL 1.1 in Overview

IDEAL provides a usable, understandable approach to continuous improvement by outlining the steps necessary to establish a successful improvement program. Following the phases, activities, and principles of the IDEAL model has proven beneficial in many improvement efforts. The model provides a disciplined engineering approach for improvement, focuses on managing the improvement program, and establishes the foundation for a long-term improvement strategy. The model consists of five phases:

I - Initiating Laying the groundwork for a successful improvement effort.
D - Diagnosing Determining where you are relative to where you want to be.
E - Establishing Planning the specifics of how you will reach your destination.
A - Acting Doing the work according to the plan.
L - Learning Learning from the experience and improving your ability to adopt new technologies in the future.

Each of the five phases is made up of several activities. The phases and activities are described below.



The IDEAL Model

return to top ▲

The Initiating Phase

Critical groundwork is completed during the initiating phase. The business reasons for undertaking the effort are clearly articulated. The effort's contributions to business goals and objectives are identified, as are its relationships with the organization's other work. The support of critical managers is secured, and resources are allocated on an order-of-magnitude basis. Finally, an infrastructure for managing implementation details is put in place.

Stimulus for change
It is important to recognize the business reasons for changing an organization's practices. The stimulus for change could be unanticipated events or circumstances, an edict from someone higher up in the organization, or the information gained from benchmarking activities as part of a continuous improvement approach. Whatever the stimulus, it can have far-reaching influence on the effort's visibility, conduct, and ultimate success, change for the sake of change rarely results in significant improvement. In general, when the business reasons for change are more evident, there is greater buy-in throughout the organization and there are greater chances for success.

Set Context
Once the reasons for initiating change have been clearly identified, the organization's management can set the context for the work that will be done. "Setting context" means being very clear about where this effort fits within the organization's business strategy. What specific business goals and objectives will be realized or supported by this change? How will it affect other initiatives and ongoing work? What benefits (such as return on investment or improved capabilities and morale) will result? Context and implications often become more evident as the effort proceeds, but it is important to be as clear as possible regarding these issues early in the effort.

Build Sponsorship
Effective sponsorship is one of the most important factors for improvement efforts. It is necessary to maintain sponsorship levels throughout an improvement effort, but because of the uncertainty and chaos facing the organization in the beginning of the effort, it is especially important to build critical management support early in the process. The commitment of essential resources is an important element of sponsorship, but effective sponsors often do much more than this. Sponsors can be most effective if they give personal attention to the effort and stick with it through difficult times.

Charter Infrastructure
Once the reason for the change and the context are understood and key sponsors are committed to the effort, the organization must set up a mechanism for managing the implementation details for the effort. The infrastructure may be temporary or permanent, and its size and complexity may vary substantially depending on the nature of the improvement. For a small effort, the infrastructure may be a single part-time employee; for a large and complex effort, such as software process improvement, it may involve 2-3% of the organization's people across a number of groups. Chartering the infrastructure involves developing explicit written agreements that document and clarify expectations and describe responsibilities.

The activities of the initiating phase are critical. If they are done completely and well, subsequent activities can proceed with minimal disruption. If they are done poorly, incompletely, or haphazardly, then time, effort, and resources will be wasted in subsequent phases.

return to top ▲

The Diagnosing Phase

The diagnosing phase builds upon the initiating phase to develop a more complete understanding of the improvement work. During the diagnosing phase two characterizations of the organization are developed: the current state of the organization and the desired future state. These organizational states are used to develop an approach for improving business practice.

Characterize Current and Desired States
Characterizing the current and desired states is similar to identifying the origin and destination of a journey. Characterizing these two states can be done more easily using a reference standard such as the CMM for Software. Where such a standard is not available, a good starting point is the factors identified as part of the "stimulus for change" activity. This activity should focus on elements critical to the changes being introduced, rather than every aspect of an organization's work.

Develop Recommendations
The recommendations that are developed as a part of this activity suggest a way of proceeding in subsequent activities. The diagnosing phase activities are most often performed by a team with experience and expertise relevant to the task at hand. Their recommendations often weigh heavily in the decisions made by key managers and sponsors.

return to top ▲

The Establishing Phase

The purpose of the establishing phase is to develop a detailed work plan. Priorities are set that reflect the recommendations made during the diagnosing phase as well as the organization's broader operations and the constraints of its operating environment. An approach is then developed that honors and factors in the priorities. Finally, specific actions, milestones, deliverables, and responsibilities are incorporated into an action plan.

Set Priorities
The first activity of this phase is to set priorities for the change effort. These priorities must take many factors into account: resources are limited, dependencies exist between recommended activities, external factors may intervene, and the organization's more global priorities must be honored.

Develop Approach
Combining increased understanding of the scope of work (gained in the diagnosing phase) with a set of priorities leads to the development of a strategy for accomplishing the work and identifying resource availability. Technical factors might include the specifics of installing the new technology and new skills and knowledge required for using a technology. Non-technical factors, including the organization's culture, likely sources of resistance, sponsorship levels, and market forces, also must be considered.

Plan Actions
With the approach defined, a detailed implementation plan can be developed. This plan includes schedule, tasks, milestones, decision points, resources, responsibilities, measurement, tracking mechanisms, risks and mitigation strategies, and any other elements required by the organization.

return to top ▲

The Acting Phase

The activities of the acting phase help an organization implement the work that has been conceptualized and planned in the previous three phases. These activities will typically consume more calendar time and more resources than all of the other phases combined.

Create Solution
The acting phase begins with bringing all available key elements together to create a "best guess" solution to address the previously identified organizational needs. These key elements might include existing tools, processes, knowledge, and skills, as well as new knowledge, information, and outside help. The solution, which may be quite complex and multi-faceted, is often created by a technical working group.

Pilot/Test Solution
Once a solution has been created, it must be tested, as best guess solutions rarely work exactly as planned. This is often accomplished through a pilot test, but other means may be used.

Refine Solution
Once the paper solution has been tested, it should be modified to reflect the knowledge, experience, and lessons that were gained from the test. Several iterations of the test-refine process may be necessary to reach a satisfactory solution. A solution should be workable before it is implemented, but waiting for a "perfect" solution may unnecessarily delay the implementation.

Implement Solution
Once the solution is workable, it can be implemented throughout the

organization. Various roll-out approaches may be used for implementation, including top-down (starting at the highest level of the organization and working down) and just-in-time (implementing project-by-project at an appropriate time in its life cycle). No one roll-out approach is universally better than another; the approach should be chosen based on the nature of the improvement and organizational circumstances. For a major change, implementation may require substantial time and resources.

return to top ▲

The Learning Phase

The learning phase completes the improvement cycle. One of the goals of the IDEAL Model is to continuously improve the ability to implement change. In the learning phase, the entire IDEAL experience is reviewed to determine what was accomplished, whether the effort accomplished the intended goals, and how the organization can implement change more effectively and/or efficiently in the future. Records must be kept throughout the IDEAL cycle with this phase in mind.

Analyze and Validate
This activity answers several questions: In what ways did the effort accomplish its intended purpose? What worked well? What could be done more effectively or efficiently? Lessons are collected, analyzed, summarized, and documented. The business needs identified during the initiating phase are reexamined to see if they have been met.

Propose Future Actions
During this activity, recommendations based on analysis and validation are developed and documented. Proposals for improving future change implementations are provided to appropriate levels of management for consideration.

return to top ▲

Future of the IDEAL Model

The explosion of technological development has led organizations to adopt new technologies at an increasing rate. The IDEAL Model provides an effective approach to adopting improved software engineering processes, methods, and tools. In this article, we have described Version 1.1 of the model at a very high level. In the future, the IDEAL model will be developed further and its applicability to additional technical areas will be tested. The IDEAL Model will be documented on the SEI Web site at and the latest developments will be available there.

return to top ▲

Acronyms

CMM : Capability Maturity Model
TAA : Technology Adoption Architecture

For more information about the IDEAL model, contact ideal-team@sei.cmu.edu

Return to top ▲ | Return to IDEAL main page

The Software Engineering Institute (SEI) is a federally funded research and development center sponsored by the U.S. Department of Defense and operated by Carnegie Mellon University.

URL: http://www.sei.cmu.edu/ideal/ideal.bridge.html
Last Modified: 9 January 2004

01137198 CMP ACCESSION NUMBER: CRN19970908S0097

IT Organizations That Focus On Service Will Win The Race

COMPUTER RESELLER NEWS, 1997, n 753, PG127
PUBLICATION DATE: 970908
JOURNAL CODE: CRN LANGUAGE: English
RECORD TYPE: Fulltext
SECTION HEADING: White Paper:Hewlett-Packard Co.
WORD COUNT: 5188

TEXT:

EDITOR'S NOTE:Finding ways to quantify the value delivered to customers is every IT providers' challenge. The following is an excerpt of a white paper from Hewlett-Packard Co., Palo Alto, Calif. For the full text, see http://www2.hp.com/openview/rpm.

--

In the early decades of its existence, the Information Technology (IT) organization of an enterprise was seen primarily as a technology provider.

In most organizations, IT acquired the necessary data-processing technology to support the business of the enterprise, housed it, maintained it and replaced it when newer technology became available to meet the ever-increasing thirst for the power electronic- information processing brought to the business. Due to the complex and mystical nature of keeping large computer systems running, the IT organization often existed as an island of technology-inhabited by a different breed of folk, living behind glass walls, doing who-knows- what, apart and very much different from those who participated in the day-to-day operation of the business.

As computer technology evolved from the mainframe to the mini to the micro, and as this technology moved from the domain of the guru to that of the ordinary citizen, corporate expectations about the IT organization changed as well. The expectation of IT being a technology provider changed to one of being a service provider, due primarily to the fact that anyone with a budget could now go out and acquire information processing technology down at the local " computermart." The focus of business users is now firmly directed at how IT can provide some value-add over simple technology acquisition (no longer a big mystery). Expectations are evolving that say IT must now be a significant contributor to the business mission of the enterprise. Coming with that expectation is a strong emphasis on bottom-line performance such as ROI, competitive advantage, and industry-leading thinking.

In reality, IT has always been a provider of service, rather than just technology. It has taken the shift in how technology is delivered to business users to make that fact apparent to those outside the glass house. No longer can IT focus on technology for the sake of technology as it was wont to do in the first 30 years of its existence. Rather, IT managers must now strongly focus their talents on the use of technology for the sole purpose of providing services to the enterprise to increase its viability in competitive markets. IT organizations that have not accepted this fact have found themselves the focus of intense scrutiny by business managers within the enterprise with oftentimes dire results such as outsourcing, forced re-engineering efforts (often on a massive scale), and the associated instability and personal trauma that accompany such alternatives.

Added to this situation has come a rapid and fairly radical shift in the basic technology used to support IT organizations within the enterprise. The movement from a centralized, small number of large systems running a common operating environment (mainframes) approach, to a decentralized model based upon numerous distributed heterogeneous systems connected by corporate networks, has just added fuel to the fires of chaos within many IT organizations. IT managers who were comfortable within the confines of the glass house, managing stable, well-understood mainframes, are suddenly faced with understanding and implementing new,

less mature (but supposedly more cost-effective) bits of technology. Add this to the challenge of becoming a responsive, quality-of-service based organization as previously mentioned, and it is no surprise that IT organizations and the people within them are facing challenges that had been forgotten or had never been experienced.

What IT managers often do not understand is that it not the technology itself that provides service to the enterprise, but rather it is the proper application of the technology. It is indeed a small , but nevertheless significant, paradigm shift to make the transition from a technology provider mentality to that of a service provider way of thinking. All the technology in the world will not make an IT organization a better service provider. What is required to successfully make the transition is a management strategy and a set of tactical methodologies that support the objective of being a world -class service provider.

This leads to the topic of Service Management or, more specifically , IT Service Management (ITSM). ITSM has been used in the IT industry for many years in one form or another, usually by organizations that have realized the necessity for a balanced, service delivery approach within IT. ITSM is based upon the philosophy of defining, achieving and maintaining required levels of IT service to the business user population with the enterprise. Unfortunately, it can be said that very few IT organizations have adopted ITSM as a key strategy toward meeting their stated objectives within the larger enterprise. Even more unfortunate is the fact that on the path toward adoption of the distributed IT model, many organizations have left behind an orientation toward service while they struggle to gain control over new and somewhat immature technologies.

In order to meet the expectation of the enterprise that IT is a significant contributor to the business only in the context of providing value-added services above technology acquisition, IT organizations in general need to (re)adopt the ITSM philosophy. Like any set of key management processes, successful ITSM requires a commitment at the highest levels of management within IT organizations. The good news is that the process is well-understood and thoroughly documented. Consequently, no invention is required- only execution.

The remainder of this paper covers the subject of ITSM in more detail. It is specifically written to reflect the needs of IT management who are working with distributed technologies and reflects some of the challenges and new ways of thinking required to successfully implement ITSM in the distributed environment. The material in this paper is derived from actual experience in consulting with IT groups that were looking to make the transition to service-based organizations within their respective companies. The material has been refined over the years to reflect the changing face of technology used within commercial IT, and has proven successful for those organizations that have adopted the processes described here.

Although the rigor of adopting and implementing new management practices may seem less than appealing at this juncture in technology evolution, it appears the alternatives may be even less appealing to those who are charged with delivering IT services to their respective companies. Failure to implement ITSM or similar practices is just delaying the inevitable-and it will only get harder as the pace of technology evolution continues to accelerate.

Background

Service Management (sometimes referred to as Service Level Management) is an activity that has long been practiced in the data center. Typically related to the effective deployment and management of mission-critical applications, Service Management is a strategy intended to provide (and, in many cases, guarantee) levels of service from IT to business users who rely on IT applications as a tool to support their mission within the enterprise.

As the global IT community has moved toward implementation of the distributed computing paradigm based upon operating environments such as Unix, Microsoft Windows, Windows NT, Windows 95 and TCP/IP-based local and internetworks, the concept of Service Management has been relegated to the background or, in some cases, ignored altogether. Based upon the history of technology evolution in recent years, this was to be expected.

Adoption of new technologies focuses almost entirely on implementation issues early in the life cycle, which have a tendency to ignore the more mundane and fundamental tasks of management. However, as IT organizations progress along the time line of technology adoption and as implementation issues are overcome, the need to effectively manage the technology to support day-to-day business objectives again becomes paramount.

It appears that we are now at that point in the time line. Early adopters of distributed IT technology have successfully deployed systems and their interconnecting networks, and have migrated or developed mission-critical applications to run on that infrastructure. The distributed computing model is now mainstream and, as such, IT managers and the industry at large are starting to focus less on firefighting (although this is still a highly visible activity) and more on "we're in it for the long haul" types of management activities needed to support the mission-critical IT environment.

The key to effective IT management is still Service Management. The problem of providing consistent, timely, high-quality service to business users in the enterprise has not changed-only the underlying technology to support it. However, this new distributed computing model brings with it challenges for effective Service Management that were overcome years ago in a mainframe-based IT organization. In fact, it seems that it was sufficiently long ago that these practices were developed that many IT managers have forgotten how they were derived, and are now at a loss to rapidly re-create that management structure now that they are faced with the same task in the distributed environment.

Service Management Is A Strategy

As technologists are wont to do, they have a tendency to become tactical very quickly in search of a solution to a perceived problem. This stems primarily from the "firefighting" mentality that goes along with the implementation of new (and sometimes immature) technology. This is not to say a firefighting mentality is a flaw in the way IT organizations operate. Quite the contrary- if we didn't have good firefighters it is highly unlikely we would have seen the rapid adoption and use of new technologies that promise to lower IT costs and make IT organizations more responsive to the business needs of the enterprise.

However, firefighting is a tactic, not a strategy. It is a reactive activity and if used as a management method, creates the classic oxymoron "reactive management." Proactive methodologies stem from a strong base of planning. This means an effort has been made to set objectives based upon a clear organizational vision. From those objectives stem a clearly defined set of strategies that can be expanded into implementation tactics to support the objectives set forth early in the planning process. This is a standard planning methodology used by organizations to successfully manage their activities throughout the world. Much can be learned from this methodology when tackling the problem of managing the distributed IT environment.

Service Management is one such strategy. It stems from the objective of the IT organization of providing a high level of service to business users within the enterprise. Words typically found in objective statements in this area include "timely," "consistent," " quality," "productivity" and "value." Strategies typically emanate from the "top down," so Service Management is a concept that is of great importance to CIOs, CFOs and strategic planners within the enterprise.

Essentially, IT Service Management is the strategy of defining, achieving and maintaining required levels of IT service to the business user population with the enterprise. The reason Service Management is such an effective strategy is because it focuses on the needs of the business user as the primary driver for the development of the IT infrastructure. Rather than arbitrarily deploying computers and networks of various capabilities and capacities, an effective Service Management strategy takes into consideration the needs of the user population for any given application area when designing and implementing that portion of the IT infrastructure. The by-products of this activity are twofold:

- A higher return on investment in IT expenditures:By using the needs of the IT customer to specify the capabilities and behavior of the IT infrastructure, costs are understood early in the cycle. Excess capacities can be avoided, and proper ongoing management activities are

understood, can be planned for, and are staffed. As such, capital and personnel costs can be better understood and controlled.

- Fewer failures through proper expectation setting: By working with the business user during requirements and planning activities, their needs are known and IT analysts can help them understand if their expectations can be met within the fiscal constraints of the IT organization. They also feel their input was considered from the beginning and they are "part of the solution," rather than "part of the problem."

Not only can Service Management be used when initially developing a distributed IT infrastructure, but it can be used to gain control over resources that may have been deployed in a more "ad hoc" manner. However, in both cases, it is necessary to adopt Service Management as a key strategy to be applied aggressively within the IT organization.

The remainder of this paper focuses on aspects of the deployment of Service Management within the IT organization, and discusses some of the current challenges that face IT organizations as they make use of this methodology.

Service Level Agreements

The first level of tactical support for Service Management is the Service Level Agreement (SLA). Service Level Agreements are documents prepared explicitly to define the various service levels IT is expected to deliver to the business user within the enterprise. As part of the overall problem decomposition process, SLAs are written to focus on individual applications and the service levels required by the application users. Therefore, it is typically necessary to prepare more than one SLA for any given business user community (e.g. , order entry, HR, etc.). In some cases where applications span multiple business users (e.g., E-mail, Internet access, etc.), an SLA is prepared to reflect the needs of heterogeneous business users, but with common service needs.

Rarely is it effective to combine multiple applications in a single SLA. One of the challenges faced by IT organizations is proper decomposition of the problem into manageable parts. Unless there are tightly defined similarities between applications servicing the enterprise (and it is not out of the question that these exist-office productivity, for example)-an SLA should be confined to focusing on a given application.

Are SLAs required for all applications? The answer is typically " No." The characteristics of applications for which SLAs are prepared usually include mission-critical applications (those which are vital to the day-to-day operation of the enterprise and without which the enterprise would suffer significant business stress), mission- sensitive applications (those whose loss or delivery of inadequate service would impact normal productivity), and those which have a high level of visibility within the organization (political). In the case of mission-critical and mission-sensitive applications, the need for Service Management is obvious. Highly visible applications, even if they are not of the mission-critical or mission-sensitive variety , can bring undue scrutiny of the IT organization if adequate levels of service are not provided, and should be considered as part of the mix of applications for which SLAs are used. Service Level Agreements have been in use in the mainframe data center environment for years. Why? Applications with the characteristics described above have typically been found running on mainframes managed by the enterprise data center. Consequently, the reader can see that SLAs are not new and their use has been proven over the course of time.

Service Level Objectives

For IT managers, the most important by-product of the SLA is the ability to accurately define Service Level Objectives (SLOs) for the IT organization against a particular application. SLOs are solely in the domain of the IT manager and his staff. They are derived from what the stated business user service levels need to be based upon the development of the SLA. From SLOs come the actual metrics IT management needs to collect, monitor, store and report on to determine if IT is meeting the agreed-upon service levels for the business application user. For example:

- An application needs to be available for use during certain shifts (availability). This requirement decomposes into days of the week, hours

of the day, etc. A metric needs to be established that reflects application availability from the standpoint of the business user. Based upon the implementation method used (single system, n- level client/server, etc.) measures need to be established and taken to determine adherence to this objective, as well as (and more importantly from a business user satisfaction standpoint), when it is not. Procedures then need to be established for the collection of this metric, as well as what to do if it falls into non-compliance (e.g., fault management). Planning needs to take place to schedule maintenance of hardware and software so as not to impact availability. Backups need to be scheduled, or alternative means to off-line backup determined, so as not to affect application availability, etc.

- An application needs to meet certain responsiveness and throughput requirements (performance). Measurements need to be taken to understand the application's behavior during normal periods of use to determine if user response time meets the service level specified in the SLA against
28 specific business transactions. If volume of work is an agreed-upon service level, business transaction counts need to be taken. As before, comparison against service levels must be made and, if exception conditions exist, have processes in place to handle the exception (performance analysis). In the planning arena, specification of required responsiveness and work volume-service levels lead to proactive activities such as capacity planning for IT infrastructure components affected by the particular application.

- An application needs to meet certain privacy requirements (security). Mechanisms must be put in place to ensure secure access to certain applications due to company confidentiality or competitive threats. Measures to the effectiveness of this system must be put in place and monitored against service objectives (e.g., no unauthorized
29 access, different levels of access to certain parts of an application, etc.). Again, processes must be in place to act upon exceptions to the service objective. As above, long-term storage of these metrics must be accomplished to accommodate reporting mechanisms to show adherence or exceptions to the service objective.

- An application must meet reliability requirements (accuracy and recovery). Somewhat related to fault management, measurements must be established and collected to determine the accuracy of an application. This may require periodic sampling of the work accomplished by the
30 application, or may require the application to be instrumented to provide those measures in realtime. If recovery is an issue, measures are needed to understand IT's ability to respond to a natural disaster or if a key database service is lost and must be re-established.

As the reader can see, Service Level Objectives are a natural by-product of the service levels determined and articulated through the Service Level Agreement process. Where the most confusion has been found, however, is in the establishment of the proper metrics to measure
31 service-objective compliance. This is especially true in the distributed environment, where the components of the infrastructure that need to be measured are many, are heterogeneous in nature (come from different vendors) and may be located in areas that are geographically remote from the center of management activity.

The Right Metric For The Job

In the early days of mainframe computing when IT Service Management was first getting its start, it was quickly realized that getting the metrics necessary to track service objectives was difficult. In some cases, the proper source of the metric was not available. In other cases, even if the metric was available, there was no consistent way to log and access the metric(s) for management purposes. IBM developed a logging
32 mechanism called Systems Management Facility (SMF) to help solve this problem. SMF provides a logging facility for IBM subsystems (CICS, DB2, RMF, VTAM, IMS, etc.) that contain instrumentation and report on their behavior. SMF also provides the application developer the ability to log data from user applications, which can then be accessed in the same way (sub)system data is accessed. This simplified the IT managers' job by providing consistency of metrics across subsystem and application boundaries. It also provided the impetus for a multimillion-dollar-a-year industry for ISVs who provided management applications around the SMF

data source.

In the Unix and NOS (Microsoft NT, Novell NetWare, etc.) environments, there is yet to exist a standard way to log management metrics that is similar to SMF, but technology now exists that overcomes this limitation. This has caused many IT managers who come from the SMF world (and who relied heavily on tools that use SMF data) consternation due to the lack of a consistent data source. Those IT managers who come from environments where a robust management data source is not available
33 are also oftentimes unaware of the benefits this type of facility can provide when administering to their systems and the applications they support. Consequently, what we see in the distributed environment is an attitude of "If I can't understand it, I can't manage it" and, as such, this has supported a slow movement toward the definition and use of SLAs and service objectives within IT organizations that support distributed computing environments based upon Unix and NOS platforms.

What needs to be acknowledged is that although SMF (and SMF-like mechanisms) have worked well in the large, proprietary environment, this philosophy breaks down in the distributed environment. This is primarily due to the large volume of data collected and the associated overhead on
34 system resources required to support the collection (disk storage, CPU to process the large volume of data, etc.). Alternatives to this approach are now starting to be used, which in essence stress collecting just the "right metric" for the management task at hand.

Based upon a clear definition of the service objective for any given service level, and having defined the proper metrics to support adherence to the service objective, management metrics can be classified in at least three different ways. Each class of metrics has a particular
35 management activity in mind as part of monitoring and maintaining service levels. The intent is to only gather the metrics necessary for a particular management activity, which reduces the overhead of collection and storage, as well as reducing the burden on management tools and human operators in the interpretation and required action on the data. Three classifications of metric collection include:

- Health and/or Service Objective Compliance:a small number (1-10) of metrics that are intended to provide a high-level view of system, subsystem or application health and service level compliance. These metrics are typically used in a realtime context in a management by
36 exception mechanism. These metrics are looked upon as indicators only , and have discrete values which may equate to a traffic light. Green -all is well. Yellow-there is trouble brewing and attention may be required. Red-a problem is occurring (or has occurred) and immediate attention is required.

In most cases, these metrics are not atomic in scope-rather they are a result of many different aspects of the behavior of the system, subsystem, network or application being analyzed. Output of the analysis are the health and SLO compliance metrics. This implies additional levels
37 of intelligence in the data-collection mechanism- more than just instrumentation, collection and logging. This intelligence is typically found in an "agent," a program/monitor running on a system at all times, charged with watching over the system and alerting a management console when anomalies occur.

Since local intelligence is used to watch over a system, and management by exception notification to a centralized console is used , network overhead is significantly reduced. Only a small message alerting
38 an operator or help-desk administrator would be needed, and the local agent would initiate the message, as well as the capability of initiating and monitoring some local action that may help alleviate the situation.

- High- to Medium-Level Problem Resolution:There are two different foci for this set of metrics. In a problem-resolution situation (yellow/red alert), a larger number of metrics (say, up to 40) are needed
39 to "drill down" into the problem. The idea is to present enough metrics to understand the problem or noncompliance situation roughly 80 percent of the time, but not overburden the human operator or agent program with excessive detail. In many cases, this data would be logged in order to give a better picture of the phenomenon that is being observed over time through the capturing of the particular set of statistics.

The second use for this set of metrics would be to support trending

and general resource models for use in management reporting and IT financial applications. In all cases this data would be logged much as it is today, but already summarized at the point and time of collection to avoid the overhead of logging excessive levels of detail.

In both cases, the data would be logged locally and kept for a short period of time (perhaps up to 30 days). Mass-storage requirements would be held to a minimum based upon the presummarized nature of the metrics being logged. Network overhead again would be minimized by transferring summary/preprocessed data to a centralized repository for longer-term data storage and centralized management and reporting.

- Detailed Troubleshooting, Optimization or Modeling Metrics:As many metrics as necessary to solve the other 20 percent of the problems or to use for an optimization/tuning or modeling exercise when doing application/subsystem development.

This is fairly close to the method used today by many metric providers but is only needed in a small number of cases and for a short period of time. There are cases when summarized or interval data just is not granular enough to solve a problem or to characterize a highly detailed model, and this classification and mode of metric generation is designed to accommodate those situations. Again, this level of granularity should be selectable by a tool or an agent so that it is only used in the cases where it is needed. The overhead of this method of collection can be quite high and will typically impact system responsiveness in some fashion. If transferred to a central location for further use and analysis, network overhead will be a factor. If this classification scheme is used, the task of collecting management metrics for SLO monitoring and problem solving is simplified significantly. Confusion is reduced as to the use of a particular set of metrics and increased focus can be given to the quality of the metrics, rather than the quantity. It also allows for a phased implementation of service level monitoring and what it means.

Monitoring is the act of determining compliance to agreed-upon service levels-not determining why they are not being met. Although important, problem resolution (as opposed to determination) is a separate, more tactically focused activity than monitoring for compliance. In normal day-to-day operations, in a well-understood and properly designed infrastructure, problems are the exception, not the rule. Consequently, in the early phases of service level management, monitoring for compliance is the first step. As problems are determined and characterized, methods for their resolution are a natural by-product of the service management problem decomposition process.

The bottom line is this-focus on getting the right metric for the management task at hand. This implies the task is understood and the desired solution has been well-defined. Ignore the urge to create a solution and then look for a problem to solve with it by collecting every piece of data, all the time, with the thought that someday it may be needed.

Service Management Support Tools

Having the right information to monitor service level compliance is the biggest step toward successful Service Management. However, without the proper support tools to help in the management task, Service Management can prove to be a daunting activity. As mentioned previously, it was essentially the creation of the SMF logging facility that enabled a broad base of management tools to appear on the scene to assist in service level management on the mainframe. Resulting tools developed to use this data made it possible for IT managers to understand and properly control their environment.

Based upon where the Unix and NOS environments are in their evolution toward mission-critical support, we are seeing more and more management support tool offerings appear. Although slow in starting, the management tool industry is gaining momentum rapidly and the rate of release of new solutions has become exponential.

For instance, less than four years ago, there was literally a dearth of system management tools (administrative, operations, etc.) available in the Unix and NOS marketplace. The management tool scene was dominated by TCP/IP network management tools designed to help the network administrator start to get control of his or her far-flung set of network

components, data channels, etc. Based primarily on SNMP (Simple Network Management Protocol) version.1, these network management tools offered centralized fault detection and control of network components and essentially paved the way for the successful implementation of TCP/IP based local and internetworks throughout the world.

Since then, there has been a surge of system management tool offerings showing up in the marketplace. Evolving from the philosophy of treating a computer system like a network component so SNMP management platforms could monitor them, we have seen operations and administration platforms developed specifically with the idea of managing systems. Alternative methods of communication between agent and consoles have evolved based upon non-SNMP mechanisms, which have been traditionally used in the network management platform tool offerings. In addition, the trend toward integrated system and network management platforms continues to increase, to allow for managing complete portions of the infrastructure from a single console.

However, there are many critics of these tool offerings-and in large part for good reasons. Due to the stage of evolution of these tools, and due to the fact that they come from many different vendors , integration of applications has been an issue. Even though we have common consoles from which to manage both our networks and systems, the management applications rarely interact well, if at all. On top of this, these tools still give a discrete component view of the infrastructure-e.g., we still look at individual systems or network components.

With the increasing emphasis on managing applications (consistent with the Service Management philosophy) rather than components, there is still a ways to go in creating suites of tools that focus on application management rather than system or network management. The good news is, however, these shortcoming are being acknowledged by the tool providers and significant work is being undertaken to get results in this area.

The better news is that current tool offerings, especially those based upon centralized frameworks that allow for control of the distributed environment, are more than adequate to perform effective Service Management. Recall the biggest challenge of Service Management is the determination of appropriate service levels and the metrics needed to monitor them. There are many tools available today, using more than one technology, that are adequate to monitor service objective compliance. In fact, most distributed management tools initially were developed to allow for generating alerts to managers letting them know a component was out of tolerance. Technology is also available for the logging and access of key management metrics for reporting and troubleshooting purposes across multiple platforms, with more platforms on the way. The tool offering in the area of distributed system and network management will continue to evolve at a rapid pace. There will be shake-outs in terms of who the dominant players are, and it seems logical than many partnerships and alliances will be formed from within the ranks of current and future players.

Heck, Michael C.

From: DeMille, Ginger R. (ASRC)
Sent: Tuesday, August 31, 2004 2:56 PM
To: Heck, Michael C.
Subject: SEI Automation Tools - General info

Perhaps you should ask Karen to obtain copies of user manuals for some of the more commonly known CASE tools - for future use.

http://www.cs.queensu.ca/Software-Engineering/vendor.html - index of CASE tools.

8/7/7 (Item 4 from file: 148)
DIALOG(R)File 148:Gale Group Trade & Industry DB
(c)2004 The Gale Group. All rts. reserv.

05910516 SUPPLIER NUMBER: 12503495 (THIS IS THE FULL TEXT)
CASE: a testbed for modeling, measurement and management. (Special Section: Computer-Aided Software Engineering in the '90s)
Tate, Graham; Verner, June; Jeffery, Ross
Communications, v29, n4, p65(8)
April, 1992

TEXT:
Human activity in the developed world strives not only to maintain status quo activities and lifestyles, but to improve on them. In particular, the application of technology has been focused on improvement. Technology is central to organized society's efforts to improve the lot of individuals and organizations, regardless of one's opinions of the success or failure of instances of technological application or of the ultimate nature of improvement.
This ethos of improvement or "doing better" has strongly influenced attitudes toward software development and maintenance. From the software crisis of the mid-1960s, well described in |6
, to the present day, many concepts, methodologies, languages, tools and techniques have been introduced with the aim of improving the software process and its products. Particular initiatives which we will examine here

are CASE and the software improvement paradigms of the Software Engineering Institute (SEI), Software Process Capability Maturity Model (CMM), |10, 13, 14
and the University of Maryland's Tailoring a Measurement Environment (TAME) project |1, 2
.
CASE aims at greater automation of software production. Just as CAD/CAM has brought integrated design tools to the engineering of physical systems, CASE is bringing analogous tools to the more abstract engineering of software. Ultimately, the motivation for tool use is economic--for competitive advantage. There are many aspects to competitive advantage, including time-to-market, productivity, quality, product differentiation, distribution and support. Software engineering, however, has a narrower scope, comprising software definition, design, production, and maintenance. CASE aims to improve these activities through the use and integration of software tools.
Software improvement has recently received more explicit emphasis, together with a firmer conceptual and empirical basis, through the work of the SEI on the CMM |10, 13, 14
and the work of Basili and Rombach on the TAME project |1, 2
. Central to both of these major research efforts has been the

characterization and improvement of the software process. There are differences in the two improvement paradigms, which will be examined briefly later, but they agree in assigning a central role to software metrics for both software process characterization and improvement.
The term software process has been used to designate the complex process by which software is developed, from conception to operation. Traditionally, the software process has been depicted informally in software development life cycle (SDLC) diagrams, usually augmented by textual descriptions. Recently, however, more formal approaches to software process modeling have emerged |12, 18
with the construction of more precise and detailed models. The computer-aided enaction of such models by developers can potentially provide a degree of definition and traceability which has been difficult to achieve in the past and should lead to the collection of much better data about software development.
The important developments in CASE, software maturity and improvement, software metrics, software process modeling and enaction, are brought together in this article and their relationships explored. The salient relationships between contributing areas are illustrated in Figure 1. A modeling, measurement and development tool architecture is presented which integrates CASE, metrics, and process model enaction and thereby provides a framework and testbed for the understanding, management, and improvement of the software process.

CASE, Software Improvement and Metrics

In view of the software improvement intent of CASE, it is appropriate to examine CASE in relation to major software improvement models, namely the CMM and the TAME model.

CASE and the SEI Capability Maturity Model

The CMM is a five-level model in which higher levels indicate greater software process maturity which is in turn associated with increasing productivity and decreasing risk. Level 1 (Initial) represents the crisis-driven, ad hoc development which is still all too common. Level 2 (Repeatable), while still intuitive and dependent on individuals, exhibits regularity in repeating previously mastered tasks and the beginnings of manageability. Level 3 (Defined) is characterized by a specified and institutionalized software process, no longer so dependent on individuals, and by having a **Software Engineering Process Group** to lead process improvement. At Level 4 (Managed) the process is measured and controlled in the sense that relationships between activities are understood quantitatively so that, for example, variations in early activities can be analyzed to determine their expected effects on later activities. Managers have a firm quantitative basis for their plans and decisions. Level 5 (Optimizing) provides not just for the management of a defined process using automatic data collection but for change and **optimization** of the process itself.
The **CMM** has been criticized on a number of counts |5
. Much of the criticism is, in fact, a tribute to its profound impact on U.S. software producers. Some authors noted that the model is strikingly indifferent to the use of technology and automation for process improvement, that organizations and projects straddle the levels, that the multihurdle grading system has statistical and methodological flaws and that the model is unproven. ("It appears unlikely that such ratings have any meaningful correlation to the actual abilities of organizations to produce high-quality software on time and within budget" |5
.) The evaluation program itself, and the importance to some organizations of getting high scores, may distort software improvement to fit the model framework. Other authors |11
have responded to these criticisms, noting that management is more important than technology, that experts have been unable to agree on the technologies necessary for software process improvement, and that the software capability evaluation instruments are evolving and subject to periodic revision. Other questions could be raised, such as the applicability of the model to application domains other than the U.S. defense domain within which it has largely been applied to date |5, 14

. It would be unrealistic to regard the model as perfect, and equally unsatisfactory not to regard the profound effect which it is having on the software community as being mainly beneficial. From our point of view, the important issues raised by the CMM are the importance of improving software capability, the establishment of an improvement framework, and the need for metrics. We also examine the role of CASE in software improvement, which CMM does not seem to address adequately. The relevance of specific CASE tools to different CMM maturity levels has been considered by |19
.
The introduction of CASE is seen by many organizations as a means of improving the consistency, repeatability and definition of their software process, as well as either the quality of their software products or the productivity of their software process, or both |17
. Since SEI maturity Level 2 is characterized by the label Repeatable and Level 3 as Defined, it can be argued that CASE would be seen by many, rightly or wrongly, as a major means of software maturity advancement to levels 2 and 3 if they were to consider their use of CASE in a CMM context. There is much work on CASE technology transfer |7, 15, 16
which indicates that CASE alone is not enough for improvement and that appropriate motivation, education and management are also necessary within a carefully planned introduction program. Nevertheless, CASE is often seen as the major technological agent on which improvement is focused.
Progress up the maturity ladder also necessitates the use of appropriate metrics to assess the effects of tools, technologies, education and management and to measure aspects of the ascent from rung to rung. The SEI CMM implies the need for software metrics in the five maturity levels as follows |14, 20
.
* Level 1, Initial: estimating software size, estimating resource needs, developing schedules, software quality assurance, gathering data on code and test errors.
* Level 2, Repeatable: need for orderly methods for tracking, controlling and improving the quality of software or of the software process; need for action plans directed at long-term software process improvement.
* Level 3, Defined: the three key challenges are all metrics related: process measurement, process analysis and quantitative quality plans.
* Level 4, Managed: both key process activities and major product properties are measured; changes are made to the software process specifications based on results of analyzing this data.
* Level 5, Optimizing: data gathering is automated; metrics have a process or process improvement focus, or their purpose is to support a technology improvement plan.
Thus, metrics is a key factor in raising the CMM software maturity level of an organization.
CASE within the TAME Framework
In the TAME project at the University of Maryland, Basili and Rombach have developed a methodology for improving the software process by tailoring it to specific project goals and environments |1, 2
. The improvement methodology is a loop with five steps:
Step 1: Characterize the environment and candidate models, methods and tools for improvement.
Step 2: Establish and quantify improvement goals and check consistency between goals and improvement agents.
Step 3: Choose first appropriate models, then appropriate methods and tools.
Step 4: Transfer the chosen technology; carry out the software project; collect, validate and analyze data, providing on-line feedback throughout.
Step 5: Do post mortem analysis and recommend new improvement goals.
In TAME, metrics are chosen using the Goal, Question, Metric (GQM) paradigm. Improvement goals lead to questions that must be answered to determine whether the goals are being achieved. These questions, in turn,

lead to metrics which are necessary to obtain quantitative answers to the questions.

In many respects TAME is complementary to CMM. From the CMM point of view, the TAME improvement methodology can be regarded as an iterative process for climbing the capability maturity ladder. From the TAME point of view, CMM is concerned with steps 1, 2 and 5 of the TAME improvement methodology: characterizing the environment, at least from a maturity level point of view, identifying key metrics issues and setting improvement goals (the next rung on the maturity ladder), doing post mortem analysis (determining what maturity level has been achieved).

Software Process Modeling, Measurement and Management

In order to measure and manage the software process, we need suitable models of it. It should be defined. Not necessarily in the extended sense in which Level 3 of the CMM is Defined, but it should be sufficiently well defined for every resource-consuming activity, and every product or component of interest, to be comprehended within the chosen model.

Traditionally, SDLC diagrams and verbal commentaries have been used to describe software development, typically illustrating variants of the well-known waterfall model and recently more sophisticated models, such as the spiral model |4
. SDLC diagrams require formalizing, however, before they can be used to define software processes, rather than just describe them. In seeking an appropriate definition vehicle, we turn to the recent research area of software process modeling.

Software process modeling has as two of its goals greater understanding and more formal description of the software process. Two significant contributions to the field have been those of Osterweil |18
and Humphrey and Kellner |12
. Osterweil's work on process programs can be summed up by the catch-phrase title of his paper: "Software Processes are Software Too" |18
. Process programming represents software development in program form, using programming languages, notations and formalisms. Such process programs, however, are not executed (since development is still far from automatic), but enacted in a symbiosis of developer, machine and process program |23
. In contrast to the essentially activity-based or functional view of process programming, Humphrey and Kellner propose a more product-based model made up of entities, such as code modules or user manuals, states, such as non-existent, undergoing development, developed, running tests, analyzing problems, passed testing, and state transitions, for example from running tests to either analyzing problems or to passed testing.

Our software process modeling goals are somewhat different than those of many software modeling researchers. We seek to measure and manage the software process. Thus the modeled process must be measurable and manageable. For these purposes, the goals of understanding and formalization are clearly helpful, but are not enough. The difference in goals imposes a number of constraints and has a number of implications for process modeling. Notations and objects must be understandable, acceptable and, if possible, natural to software managers and developers. The granularity of activity and product decomposition must not be so coarse as to bury essential detail, such as embedded rework, or so fine as to be a measurement nuisance to developers and an unacceptable overhead. Both process programming and entity process modeling are sufficiently flexible to model software products and processes at any desired level of generality or detail. They are both, however, rather formal means of representation not intended for, nor suited to, ease of communication. During enaction, a process program can present a friendly face to the developer using appropriate dialogues, menus and graphics, but this is a different matter than communicating the structure of a process model. An adaptation of systems analysis methods is more suitable for this purpose. In this approach, data flow diagrams (DFDs), or equivalent representations, are enacted. DFDs have the advantages of being widely known, easily understood, suited to the description of asynchronous person-machine systems, and

covering activity breakdown levels appropriate to developers and managers. It might be argued that DFDs are themselves, like SDLCs, too informal for enaction. However, work has been done on the formalization of DFDs for their direct execution |8
which is analogous to enaction.

The Role of Process Model Enaction

This section examines the role of process model enaction, first from a conceptual and then from a practical point of view. Conceptually, process model enaction provides a mechanism by which process models can be used for measurement and management. A software process model can be used not only to define, but also to structure, direct and record software development by having it enacted, partly by the developer and partly by the workbench on which the model is implemented. Enaction of a suitable model has great advantages for both measurement and management. If the model is clearly understood and agreed on by developers, its enaction can provide a framework within which they can work. Their activities can be clearly identified, and resource use can be recorded and directly related, at or near the time of use, to specific model activities, products and components. A suitable software process model can provide a comprehensive framework for development, and enaction can provide traceability through the model of developer effort and its effects. In concept, the role of process model enaction is a central one and its potential advantages are considerable.

From a practical point of view, enaction experiments relevant to our purposes have been conducted by Boehm and Belz |3
and by one of the authors. Boehm and Belz constructed and enacted a process program of the spiral model. Their observations are noteworthy, including ". . . the process programming experiment led to a revised process program better reflecting the realities of Spiral Model application. In particular, it identified the importance of early process requirements, architecture and design activities and the appropriateness of the Spiral Model risk-driven approach in guiding these activities." They identified many unresolved issues in the enaction of process programs but, more important, they demonstrated the feasibility of enacting a process program.

Work by one of the authors on the enaction of process models also confirms the feasibility of enaction with two different process models, at least for small development projects in closely controlled environments. The first of two enaction models tested is shown in Figure 2. It is a simple waterfall model, with feedback flows, depicted as a top-level DFD. In practice, most of the DFD processes would be exploded to a further level of detail. The requirements and design processes would both include validation or verification steps, for example. The enacting developer first chooses an activity; then he or she choose appropriate data flows into that activity and particularizes them to specific products and components. The activity is then enacted and if it results in product changes, output flows are generated. For the simple developments chosen, all activities and products were covered by the model, all effort was recorded by activity and product/component and traffic along all data flow paths measured. The second enaction model tested was a product-based model shown in Booch diagram form in Figure 3. In this model, the developer selects a product, a specific product component, and then an operation on that component, for example "develop" or "validate/verify." Effort is recorded by the activity package and the resulting product change by the product/component package. The time and date of development will normally indicate if it is enhancement or rework. A process program has been written for this model and its enaction walked through. Once again the model was adequate for a simple development within a controlled environment.

In contrast to the spiral model |3, 4
, the two models depicted in Figures 2 and 3 are somewhat passive data collection models rather than more proactive management models. In fact, they can be regarded as different views of the same underlying product-based software process model for which a process entity model can also be constructed. The model has similarities to ISTAR |9

in terms of its product/component focus and the work contract implied in the component/activity relationship.

CASE and Metrics

As noted earlier, metrics are an essential component of both the SEI and TAME improvement paradigms and Level 5 of CMM has automatic data collection as a prime characteristic |14
. In the past, the effective collection of software metrics has proven difficult and expensive. Data collection has been seen as a costly overhead and a nuisance to developers who hate filling in time sheets and do not want to be bothered with recording their activities in detail. Unpublished expert estimates of metrics collection overhead costs for different organizations and purposes have variously been 8--25% (general research), 10--20% (quality), 3--5% (progress/quality), 1.5% (progress). CASE, particularly integrated CASE (I-CASE) or component CASE (C-CASE), offers the potential to collect measurements of on-workbench activity and products automatically, both more accurately and in greater detail than existing, largely manual methods.

It is important to distinguish between automatically collectable data and data which, by its nature, must be collected manually. The metrics data of interest include effort (related to purpose, activity and product), product/component size and complexity, and quality data, such as defects and faults. In principle, it is a straightforward matter to measure the size and complexity of all products stored in a CASE repository |21, 22 once one has decided on suitable, objective metrics. The automatic collection of effort and defect data is more difficult, however.

On-workbench effort can be measured in terms of session lengths (allowing for periods of in-action), keystrokes, mouse clicks and CASE transaction counts (such as add an entity, add a relationship, delete a data type). On-workbench effort can also be related to product changes by comparing the before and after states of affected product components. However, there is much related off-workbench activity which, by its nature, cannot be recorded automatically. This includes thinking, discussion, meetings and other miscellaneous activities. Similarly, defect and fault data arise in many different situations, such as inspections, verification, use of intermediate products in later development, testing and operation, many of which may not be directly related to workbench activity. The collection of off-workbench data is an important issue which is considered in the next section.

Suitably instrumented I-CASE, or C-CASE including a metrics component, are ideal for automatable data collection and, indeed, open new measurement possibilities |21
. Much of the required data, including product/component size and complexity, can be collected by a suitable metrics component which has read access to the CASE repository. However, keystroke, mouse click and transaction counts are best collected by CASE tools themselves. There has been some discussion, particularly at the CASE '89 Workshop in London, about whether or not CASE vendors should include metrics in their products. Many users at the workshop felt they should. Vendors, however, pointed out that users were not agreed about precisely what metrics they wanted.

CASE raises new measurement possibilities and requirements. For example, in the early stages of CASE development much use is made of diagrams for system modeling and design, such as DFDs, data models and structure charts. Automatic measurement of the size and complexity of such diagrams, given suitable metrics, would clearly be very useful for estimating downstream CASE development effort. Maximizing the automation of metrics with CASE should help to reduce data collection and analysis costs, increase the accuracy and consistency of size and complexity data and provide comprehensive size and complexity trend data over time. The authors are of the opinion that automatic collection of software development data, where this is possible, should not wait until Level 5 of the CMM but, in view of the importance of metrics to all CMM levels above the first, should start as early as practicable.

CASE as a Testbed for Modeling, Measurement and Management

We now need to put all the pieces into an integration model as shown

in Figure 4. In this model the developer does not interact directly with the CASE tool set, or integrated CASE development environment, he or she is using. Instead, the developer enacts a suitable process model, such as that of Figure 2 or Figure 3. It is simplest conceptually to consider enaction as occurring in sessions in which a developer selects one component and one type of operation. Single-purpose sessions of this nature may be quite long (e.g., developing a data model); whereas others may be short and run together with other sessions (e.g., making consistent changes to both a data model and a DFD, working on each alternately). The single-purpose session is important as a measurement unit. Enaction of the software process model associates the session for measurement purposes with a specific model product/component and operation (e.g., "develop data model" or "test customer data entry/change module").

Before passing the developer on to the CASE environment, the software process model activates a metrics envelope. This takes a before-session snapshot of the repository, turns on a clock and initializes counts for on-workbench developer activity. The developer then enters the CASE development environment to which most of his or her effort will normally be devoted. At this early stage no inbuilt constraints have been implemented which would restrict the developer in such a way that he or she could only access the selected product/component and uses only the appropriate CASE transactions for the selected operation. However, CASE repository snapshot comparisons and transaction traces can be used as a check. We are relying on developer education and discipline, which is acceptable in a research environment. When the developer indicates the end of a session, the entry process to the CASE environment is essentially reversed as we back out through the metrics envelope to the software process model. An "after"-repository snapshot is taken, normally just of the selected product/component, and changes from "before" noted; the clock is switched off, the elapsed time and counts are passed to the software process model, and control also transferred to the software process model. The measurements are logged to the appropriate activity before the developer begins another session and the cycle is repeated.

The process model thus records all product/component changes as well as on-workbench activity and effort. There are several possible ways of collecting off-workbench data. One approach is to present the developer with suitable menus requesting details of off-workbench activities at appropriate intervals, say once or twice a day. Another is to request details of any related off-workbench activity at the beginning of each session. Related design reasoning, or other relevant information, can also be captured, if desired. A third approach is to attempt to record all data within the software process model framework, so that no development is done, either on or off the workbench, except within the enaction of some software process model activity. If this approach is adopted, the model is always entered by the developer when starting any work on the project and exited only when that work is completely finished. In all cases, the scope of software process model activity must be clearly defined. It must be clearly understood what activities are included and what, if any, are excluded. If high-quality data is to be collected, it is important both that data be recorded when it is recent and clear in developers' minds and that it be recorded within the same environment, and using the same conventions, as that in which it will be used.

Figure 4 shows a project manager interaction with the software process model as well as a developer interaction. This is intended to allow for associated project management activities, such as making estimates, setting and modifying goals, and monitoring progress. For example, the sizes of all products, such as requirements, data models, DFDs, database schemas, action diagrams or structure diagrams, can be continually monitored, as can the complexity of structure diagrams. A trace of product size and complexity changes over time can give a good picture of progress and productivity throughout each product's life cycle, including development, rework, enhancement and maintenance. The software process models of Figures 2 and 3 do not include project management activities at this stage. That is the

subject of a future study.
The integration model relates to parts of Steps 3, 4 and 5 of the TAME improvement methodology as follows. For Step 3: Appropriate software process models and methods can be built using DFDs, process programs, or some equivalent notation. CASE tool sets or environments can be chosen as appropriate tools. The integration model can be used in Step 4 to "carry out the software project; collect, validate and analyze data, providing on-line feedback throughout." Finally in Step 5, the comprehensive metrics collected can be used in the post mortem analysis. In relation to the CMM, the integration model can provide process definition as well as the metrics necessary to resolve improvement issues at all levels of maturity.

Conclusions

CASE is viewed by many organizations as an important agent of software improvement, but it is not sufficient on its own. When integrated with metrics and a suitable software process model, CASE can form the core of a testbed for modeling, measurement and management of the software process. It can do this for the following reasons: 1) For modeling, including empirical model validation and improvement, because CASE tools are employed within, and only within, the enaction of a suitable software process model. 2) For measurement, because the CASE tools are embedded in a metrics envelope which maximizes automated data collection and, in conjunction with the process model, provides immediacy for nonautomatable data collection and traceability for all measurement. 3) For management, because the integrated model can provide comprehensive progress reporting and can also provide data from which earlier and later activities and products can be related for the purposes of estimation, planning and monitoring.
The integrated model fits well with influential software improvement paradigms, such as CMM and TAME, providing a mechanism or platform which can meet their process definition and metrics needs. The feasibility of the integrated model has been established in a limited, controlled research environment. Much more work needs to be done, however, before it can be applied more widely in practical situations.

References

1. Basili, V.R., and Rombach, H.D. Tailoring the software process to project goals and environments. In Proceedings of the Ninth ICSE (Monterey, Calif. Mar. 30-Apr. 2, 1987), ACM, N.Y., pp. 345-357.
2. Basili, V.R., and Rombach, H.D. The TAME Project: Towards improvement-oriented software environments. IEEE TSE 14, 6 (June 1988), 759-773.
3. Boehm, B.W. and Belz, F. Applying process programming to the spiral model. In Proceedings of the Fourth International Software Process Workshop, C Tully, Ed., (Moretonhampstead, Devon, UK, May 1988), pp. 11-13. Reprinted as ACM SIGSOFT Softw. Eng. Not. 14, 4 (June 1989), 46-56.
4. Boehm, B.W. A spiral model of software development and enhancement. IEEE Comput. (May 1988), 61-71.
5. Bollinger, Terry B., and McGowan, Clement A. Critical look at software capability evaluations. IEEE Softw. (July 1991), 25-41.
6. Brooks, Frederick P. Jr. The Mythical Man-Month. Addison-Wesley, Reading, Mass., 1975.
7. Corbitt, G.F., Norman, R.J., and Butler, M.C. Assessing proximity to fruition: A case study of phases in CASE technology transfer. Int. J. Softw. Eng. Knowl. Eng. 1, 2 (June 1991), 189-201.
8. Docker, T.W.G., and Tate, G. Executable data flow diagrams. In Software Engineering 86 D. Barnes and P. Brown, Eds. IEE Computing Series 6, 1986.
9. Dowson, M. ISTAR--An integrated project support environment. In Proceedings of the ACM SIGSOFT/SIGPLAN Symposium on Practical Software Development Environments (Dec. 1986), pp. 27-33.
10. Humphrey, W. Characterizing the software process: A maturity framework. IEEE Softw. (Mar. 1988), 73-79.
11. Humphrey, W.S., and Curtis, B. Comment on 'A Critical Look', IEEE Softw. (July 1991), 41-46.
12. Humphrey, W.S. and Kellner, M.I. Software process modelling: Principles of entity process models. In Proceedings of the 11th ICSE

(Bittsburgh, Pa, May 1989), pp. 331-342.
13. Humphrey, W.S., Kitson, D.H., and Kasse, T.C. The state of software engineering practice: A preliminary report. In Proceedings of the 11th ICSE (Pittsburgh, Pa, May 1989), pp. 277-288.
14. Humphrey, W.S., Snyder, T.R., and Willis, R.R. Software process improvement at Hughes Aircraft. IEEE Softw. (July 1991), 11-23.
15. Merritt, P. CASE and culture--Observations on technology transfer. CASE '88 Advance Working Papers, International Workshop on CASE, Inc., (Cambridge, Mass. July 1988), 22: 14-16.
16. Norman, R.J., Corbitt, G.F., Butler, M.C. and McElroy, D.D. CASE technology transfer: A case study of unsuccessful change. J. Syst. Manage. (May 1989), 33-37.
17. Norman, R.J., and Nunamaker, J.F. CASE productivity perceptions of software engineering professionals. Commun. ACM 32, 9 (Sept. 1989), 1102-1108.
18. Osterweil, L. Software processes are software too. In Proceedings of the Ninth ICSE (Monterey, Calif., Mar. 30-Apr. 2, 1987), ACM 2-13.
19. Pfleeger, S.L. Process maturity as framework for CASE tool selection. Inf. Softw. Tech. 33, 9 (Nov. 1991), 611-615.
20. Pfleeger, S.L. and McGowan, C. Software metrics in the process maturity framework. J. Syst. Softw. 12 (1991), 255-261.
21. Tate, G. and Verner, J.M. Software metrics for CASE development. In Proceedings IEEE COMPSAC (Tokyo, Japan, Sept. 1991), 565-570.
22. Tate, G. and Verner, J.M. Approaches to measuring the size of application products with CASE tools. Inf. Softw. Tech. 33, 9 (Nov. 1991), 622-628.
23. Tully, C., Ed. Representing and enacting the software process. In Proceedings of the Fourth International Software Process Workshop (Moretonhampstead, Devon, UK, May 11-13, 1988), pp. 3-4.
CR Categories and Subject Descriptors: D.2 |Software Engineering
: D.2.2 |Software Engineering
: Tools and Techniques--computer-aided software engineering (CASE);
D.2.8 |Software Engineering
: Metrics; D.2.9 |Software Engineering
: Management
General Terms: Management
Additional Key Words and Phrases: Maturity model, metrics envelope, software capability, software maturity, TAME
About the Authors:
GRAHAM TATE is a professor and head of the newly established Department of Information Systems at City Polytechnic of Hong Kong. His main research interests are in information systems management and economics, software metrics and CASE. Author's Present Address: Department of Information Systems, City Polytechnic of Hong Kong, 83 Tat Chee Avenue, Kowloon, Hong Kong
JUNE VERNER is a senior lecturer in information systems at the University of New South Wales, Sydney, Australia. Her research interests include information systems management, software metrics and CASE. Author's Present Address: School of Information Systems, University of New South Wales, P.O. Box 1, Kensington, NSW 2031 Australia
ROSS JEFFERY is a professor and head of the School of Information Systems at the University of New South Wales. He has worked with many organizations in Australia and elsewhere developing cost estimation methodologies and toolsets. Author's Present Address: School of Information Systems, University of New South Wales, P.O. Box 1, Kensington, NSW 2031 Australia.

****

Electronic Information Center 3600
Phone: (703) 305-5774
Fax: (703) 306-5758
ginger.demille@uspto.gov

5/9/1 (Item 1 from file: 15)
DIALOG(R)File 15:ABI/Inform(R)

01977401 48141311

Process performance measurement system: A tool to support process-based organizations

Kueng, Peter

Total Quality Management v11n1 PP: 67-85 Jan 2000 CODEN: TQMAED ISSN: 0954-4127 JRNL CODE: TOQ
DOC TYPE: Periodical; Feature LANGUAGE: English RECORD TYPE: Fulltext
LENGTH: 19 Pages
SPECIAL FEATURE: Chart
WORD COUNT: 8224

ABSTRACT: In order to gain long-term competitive advantages many companies are focusing on business processes wherein functional organizations are replaced by process organizations. It is argued that improvement of business processes, radically or stepwise, is essential and should be supported by a holistic process performance measurement system (PPMS). Based on the literature and practical experience - gained through a 2-year project with four enterprises - a framework for conceptualizing and developing a PPMS is presented and lessons learned are outlined.

TEXT: ABSTRACT In order to gain long-term competitive advantages many companies are focusing on business processes wherein functional organizations are replaced by process organizations. In this paper, it is argued that improvement of business processes, radically or stepwise, is essential and should be supported by a holistic process performance measurement system (PPMS). Based on the literature and practical experience-gained through a 2-year project with four enterprises-a frame--work for conceptualizing and developing a PPMS is presented and lessons learned are outlined.

Introduction: Traditional measurement approaches

In today's competitive business world, enterprises must continually improve the quality of their products and services to stay ahead of the competition. During the past few years many organizational efforts have been undertaken by modern enterprises, wherein the concept of a process-centred company has received a lot of attention. In this context, assessing process performance is essential because it enables individuals and groups to assess where they stand in comparison to their competitors. In addition, assessing process performance provides the opportunity of recognizing problems and taking corrective action before these problems escalate. Although business process re-engineering, business process improvement, process management and many other similar terms have been used for almost 10 years, most enterprises do not have an integrated, holistic system of gauging their business process performance on a regular basis.

Despite dramatic changes in the business environment, performance measurement systems have been affected only marginally: both top management and lower management assess enterprise performance mainly through financial measures-non-financial aspects such as customer satisfaction or job satisfaction still play a modest role (Eccles, 1991; Sinclair & Zairi, 1996, p. 375). Some other shortcomings of traditional performance measurement systems are given by Zairi (1996, p. 18), e.g. the failure to 'do more of the same', the failure to relate performance to the process and the failure to distinguish between control versus improvement.

In order to overcome these weaknesses several propositions have been made and different approaches and techniques have been implemented in the last few years:

* Balanced sorecard: The balanced scorecard (BSC), proposed by Kaplan and Norton, is a strategic management instrument: (1) to clarify and translate

vision and strategy;

(2) to communicate and link strategic objectives and measures; (3) to plan, set targets and align strategic initiatives; and (4) to enhance strategic feedback and learning (Kaplan & Norton, 1996, p. 10). The BSC supplements traditional financial measures with three additional perspectives: the customer, the internal business process, and the learning and growth perspective. It is supposed to be a tool for describing an organization's overall performance across a number of measures on a regular basis. An important characteristic of BSC is that the tool is focused on corporations or organizational units such as strategic business units, not on business processes. It looks at business processes only as far as they have a great impact on customer satisfaction and achieve an organization's financial objectives (Kaplan & Norton, 1996, p. 27).

* Self-assessment: The roots of so-called self-assessment can be seen in the quality movement which started in Japan. In the year 1951 Japan awarded the first quality-- driven enterprises with the so-called Deming Application Prize. Encouraged by Japanese success, the US launched the Malcolm Baldridge National Quality Award (MBNQA, 1999) in 1988. Finally, the European Foundation for Quality Management (EFQM, 1999) followed in 1992 with the European Quality Award (EQA). In order to carry out self-assessment, an appropriate framework is needed. Since the award criteria of the MBNQA and the EQA are generic and well documented, they serve most often as the model for self-assessment. According to Hakes (1996), self-assessment offers various benefits: (1) it produces an objective identification of current strengths and areas for improvement; (2) it provides a useful analysis of an organization's capability, which is of real interest to potential customers; (3) it helps to create a vision in order to counter an organization's tendency to skip from one initiative to the next. Overall, self-assessment is predominantly used for strategic management and action planning, or as a basis for improvement projects.

* Workflow-based monitoring: During the last few years workflow systems have been given considerable attention, both in research and in practice. Workflow systems (this category encompasses both workflow management systems and workflow applications) support automatic or semi-automatic execution of process instances, coordination between process activities and the communication between process actors. As a byproduct of this support, masses of data are gathered. They can be evaluated automatically and may provide useful information regarding activity-related costs, queuing time of process instances, workload of process participants, etc. While traditional control covers the firm in its entirety, workflow-based monitoring is concentrated on business processes. A further difference lies in the time period reported. While traditional control offers a post-hoc view, workflow-based monitoring has the character of real-time reporting (McLellan, 1996). The merits of workflow-based control lie in the prompt reporting procedure as well as in its focus on business processes. However, it is restricted in that qualitative performance data and performance data about activities carried out manually cannot be taken into consideration. A technical view is given by the Workflow Management Coalition, who define the term 'workflow monitoring' as ". . . the ability to track and report on workflow events during workflow execution. Workflow monitoring may be used, for example, by process owners to monitor the performance of a process instance during its execution" (WfMC, 1999, p. 56).

* Statistical process control: According to juran and Gryna, statistical process control (SPQ can be defined as ". . . the application of statistical methods to the measurement and analysis of variation in any process" (Juran & Gryna, 1993, p. 377). The main objective of SPC lies in the achievement of stable processes through a reduction of process variation. Stability in a process, i.e. a state of statistical control, makes it possible to predict the behaviour of a process. Making reliable predictions regarding product quality (i.e. predicting whether the product specifications will be met) has become an important tool of competition (Juran & Gryna, 1993, p. 380).

Besides the approaches mentioned, various management instruments and tools are being applied nowadays, e.g. Activity-Based Costing systems (Cooper & Kaplan, 1991; Glad & Becker, 1996; Cooper & Kaplan, 1998) which assign more accurately than traditional instruments the costs of an organization's activities to its products and processes; the Capability Maturity Model (Paulk, 1995; **SEI** , 1998) which provides organizations with guidance for rating software process improvement programs; and ISO 9000 certification which somewhat guarantee that enterprises execute their business processes in a specified and controlled way.

The reader may ask "Which measurement system is appropriate?" "It depends", one is tempted to reply. Taking into account the view that a modern performance measurement system should support a process-oriented view, companies need a system which fulfils two requirements: first, the measurement system should be focused on processes, not on whole organizations or organizational units; second, the measurement system should evaluate performance holistically by measuring quantitative aspects as well as qualitative aspects. From Fig. 1 we can see that none of the previously mentioned approaches fulfil the two criteria given. From this we conclude that a new measurement approach is needed, an approach which helps a company to establish a process-based organization where the resources are allocated to a process owner or process manager. In such an organization, it will be crucial for the process manager to have a tool which is able to assess process performance in an integral way.
The process performance measurement approach presented herein is based largely on the business process improvement (Harrington, 1991) and business process re-engineering approach (Hammer & Champy, 1993), the balanced scorecard (Kaplan & Norton, 1996), the goal question metric (GQM) paradigm (Basili & Rombach, 1988), on the criteria of the Malcolm Baldridge Award (MBNQA, 1999) and the EFQM model (EFQM, 1999), the capability maturity model (Paulk, 1995), the goal-based process modelling approach (Kueng & Kawalek, 1997) and the stakeholder theory; see, for example, Preston and Sapienza (1990) or Jones (1995).

Figure 1.

It seems appropriate at this point to clarify three important terms: process, process team and stakeholder. According to the International Organization for Standardization (ISO), a process can be characterized as a set of interrelated resources (e.g. personnel, finance, IT facilities, equipment, methods) and activities (working steps) which transform inputs into outputs (ISO, 1994, p. 2). A process can be seen as a system or a subsystem consisting of interrelated components that have a common purpose and share a set of goals. The terms process and business process are used interchangeably. A process team can be defined, according to Brannick and Prince (1997, p. 4), as two or more people who cooperate to achieve specified process goals. Usually, team members (sometimes referred to as process agents or actors) carry out distinct activities, such as-using the example of Brannick and Prince-the surgeon and the anaesthesiologist in the operating room. In practice, process teams usually consist of a process manager and his/her subordinates, but other coordination mechanisms may be appropriate as well. The term stakeholder is -used according to Mendelow: "stakeholders of an organization are those who depend on the organization for the realization of some of their goals, and in turn, the organization depends on them in some way for the full realization of its goals" (Mendelow, 1983, p. 128).

The concept of a process performance measurement system

Imagine an athlete or an athletic team which does not measure its performance. What would happen? They could not motivate themselves, they could not judge their training methods, and they certainly would not be able to achieve world-class standards. (This analogy has been taken from Bieri (1995).) A corporation which is implementing a process-based organization based on process teams is in a similar situation. As the process teams want to be the best in their class or even on a world-class level, they are always interested in two questions: Is the current

performance of the business process better than yesterday's? To what degree are the target values fulfilled? In order to answer these two questions, a so-called process performance measurement system (PPMS) is needed.

A PPMS can be characterized as an information system which:

(1) gathers-through a set of indicators-performance-relevant data of one or several business processes;

(2) compares the current values against historical and target values;

(3) disseminates the results (current value, target value, gap and trend for each selected indicator) to the process actors; cf. Fig. 2.

After presenting this short conceptual view, we turn to a more business-oriented point of view and ask: What main advantages can the setting up of a PPMS provide? We see the following aspects:

* To communicate to employees the general direction in which a company wants to go, it has become common for top management to write down its vision and mission, then place these in visible locations. But as Chang and Young remark, "vision and mission statements do a good job of providing overall guidance and direction. However, good measures provide the operational definitions everybody can clearly understand" (Chang & Young, 1996, p. 5). Therefore, a PPMS may offer substantial help in communicating the goals and objectives of business processes. Such information can enhance inter-process and intra-process collaboration and learning as the relationships between various goals and indicators become more apparent.

If it is not understood how the business processes perform, it will be extremely difficult to figure out how they can be improved. There is no doubt that one can agree on this general rule. Some authors mention that a performance measurement system should not only serve as a warning flag about performance problems, it should also communicate the reasons for the problems (e.g. Rose, 1995, p. 64). Though this conclusion seems valid, an implementation would be difficult. As the number of potential reasons for a problem may be huge, the number of performance indicators being measured would become enormous. Therefore, it must be clearly stated that a PPMS cannot tell a manager-based on the observed level of performance-what action he or she should initiate. However, a PPMS can direct attention to relevant facts that were, in the absence of a PPMS, barely visible.

Today's companies have a large and sophisticated set of information systems to support various activities: accounting information systems, environmental reporting systems, human resources information systems, manufacturing information systems, management information systems, etc. Through the use of the many different information systems, many reports are produced. Thus, many leaders, process managers and employees are overwhelmed by the amount of data they receive. Through the deployment of a PPMS the number of traditional reports can be reduced significantly as a PPMS offers process-centred and goal-directed information.

About 10 years ago a vivid debate was launched regarding the relationship between IT investments and financial performance. Several studies showed that higher levels of investments in IT did not lead to productivity gain; a phenomenon these authors called "the productivity paradox" (Brynjolfsson, 1993). Today there is some consensus that IT's impact cannot be assessed at an enterprise level-instead, it must be measured against its support for the process goals (Mooney et aL, 1996). Since the performance measures of a PPMS are linked to the process goals, the deployment of a PPMS can effectively help to assess the impact of IT investments.

Figure 2.

Traditional performance measurement systems (e.g. financial information systems) belong to the accounting department. Information produced is provided to managers of various sections and levels, e.g. the chief

financial officer, the marketing manager, the manufacturing manager, etc. Information is then forwarded by these managers to their subordinates. A PPMS, since it is focused on business processes, has the potential to change the flow of information. Process owners and process actors can be addressed directly. Information is not filtered by each manager's individual criteria; instead, information provided is tailored by the process teams according to their process goals and needs. Moreover, changed information flow combined with process-centred information may help to form a common understanding of goals, tasks and organization of a business process. To this end, a PPMS can be seen as a tool to shape a shared mental model of a process team.

In summary, the main objective of a PPMS is to provide comprehensive and timely information on the performance of business processes. This information can be used to communicate goals and current performance of a business process directly to the process team, to improve resource allocation and process output regarding quantity and quality, to give early warning signals, to make a diagnosis of the weaknesses of a business process, to decide whether corrective actions are needed and to assess the impact of actions taken.

Process performance-from a stakeholder's point of view

While discussing performance measurement three aspects are mentioned regularly: cost, time and quality (cf. Kitchenham, 1996, p. 64). Elsewhere it has been argued that process performance should be measured in terms of quality, effectiveness, efficiency, timeliness, costs, etc. We believe that process performance measurement should not be focused on these generic concepts but rather on those people who have an interest in the business process, in others' words, a stakeholder-driven performance measurement. What does this mean? First, stakeholders of a process have to be identified. Second, for each stakeholder or group of stakeholders, process-relevant goals have to be identified. Looking at it from another angle, we may say that for each indicator to be measured one should be able to designate a group of persons who have (a) a 'legal' interest in getting information on the performance of the process, or (b) are able to improve process performance through their work.

Based on the stakeholder-driven approach, we use the term process performance as 'the degree of stakeholder satisfaction'. Using some criteria given by Gillies (1997, p. 4), the notion of performance can be characterized as follows:

* Performance is not absolute. Its meaning varies for different processes, e.g. performance of an ordering process is hardly comparable with the performance of a surgical process in an operating theatre.

* Performance is multidimensional. As performance has many contributing factors, it cannot be gathered and assessed by a single indicator.

* Performance indicators are not independent. Most performance indicators stand in a relationship with one another. For the most part, the type of relationship is either conflicting or complementary; independence is the exception rather than the rule.

The principal process stakeholders we are looking at are the four following: investors (e.g. shareholders in the profit sector, government in the not-for-profit sector), employees, customers (suppliers and buyers) and society (cf. EFQM, 1999). Since process performance is measured according to the degree of stakeholder satisfaction, each group of stakeholders will be represented by an aspect or dimensions of performance. Thus, the aspects of performance we are looking at are the following: financial aspects (to measure the degree of satisfaction of the investors); employee aspects; customer aspects; and societal aspects. In order to satisfy the four stakeholder groups in the long term, business processes need continuous improvement. Therefore, a fifth aspect has to be added-innovation; cf. Fig. 3. These aspects will now be discussed in more detail.

Financial aspects

Business processes are the driving forces of any organization (Hammer & Champy, 1993). Since processes require financial and non-financial resources and create value for the customer, they definitely have an impact on the financial situation of an enterprise. Therefore, it is obvious that a PPMS has to take financial aspects into account. This aspect has been discussed at length in various papers (e.g. Cooper & Kaplan, 1991; Eccles, 1991).

Employee aspects

In a large-scale Europe-wide survey of 12 500 employees conducted in 1992, 42% of the workforce thought that professional activity could affect their health (Anderson, 1996, p. 328). Though these numbers vary between sectors (in general, white-collar workers are less frequently affected than blue-collar workers), the result of the study emphasizes that workforce health and quality of working life should be treated as an important aspect, and therefore should be incorporated into a sound performance measurement system. Moreover, low job satisfaction can affect the behaviour of a person in dealing with the work environment. For instance, a person who is stressed may become angry and this can lead to unsatisfactory customer service. This, in turn, could discourage the customer from making further purchases and would therefore have a negative financial impact.

Figure 3.

Employee aspects may cover a broad range of subjects, e.g. communication, job conditions, physical discomfort, psychological well-being, workload, supervision, opportunities for growth, or socialization. Since the development of a job satisfaction scale is a timeconsuming task, it may be worth evaluating existing scales, such as the job descriptive index, the Minnesota satisfaction questionnaire, or the job diagnostic survey (see, for instance, Spector, 1997).

Customer aspects

"Managers cannot know how good their services are until they ask the customers" (St. Clair, 1997, p. 127). Though this perception is not completely new, practical experience shows that customer aspects are rarely systematically evaluated. This observation not only holds for enterprise-external customers, but also or even more so for enterprise-internal process customers, as they are often less visible than the former.
To evaluate process performance from the customers' point of view, two slightly different approaches exist. In the first approach the customers' expectations are compared with their perceptions. A popular instrument based on this approach is called SERVQUAL (Parasuraman et al., 1988). This instrument consists of two sections of questions: section one addresses customer expectations, section two is dedicated to gathering customer perceptions regarding a certain service they 'consume'. More recently, the originators of the SERVQUAL model have proposed several question/answer schemes in order to increase reliability and efficiency of customer satisfaction measurement (Parasuraman et al., 1994). A second approach proposes to abstain from an explicit inquiry of customer expectations. Instead, quality criteria are defined and customers have to rate both the degree of fulfilment and the importance of each criterion (see Kristensen et al., 1992).

A common characteristic of the two mentioned approaches is that they are questionnaire-- based. It may be worth keeping in mind, however, that further instruments exist, such as mystery shopping, focus group interviews, or customer advisory panels (cf. Berry & Parasuraman, 1997, p. 67).

Societal aspects
Business processes are not executed in an isolated world where only process managers, process actors and process customers interact. These business

processes are embedded in a society whose participants, or part of them, take an interest in process performance. In almost every business sector we find examples where this aspect has been neglected. Thus, a business process manager should know, for instance, what impact his or her process has on the local economy, how process-related pollution (noise, waste, etc.) is perceived by the society, what amount of energy the business process requires, whether raw material or semifinished products purchased by the company were produced under legal and humane circumstances, whether steps towards preservation of global resources have been undertaken, etc. (cf. Hakes, 1996, p. 125). Overall, measuring societal aspects of a business process means both measuring the impact a process has on its society and measuring how the impact is perceived.

Innovation aspects

Business process re-engineering may lead to high-performance processes. However, if processes are not permanently adapted, process performance will decrease in relation to that of competitors. In order to counter such a performance decline, innovations are needed. Since new ideas are the result of human creativity, it is essential that each process actor become active in learning (through personal experience, workshops, literature, etc.) and information sharing. Furthermore, selected members of a business process could be charged with the task of scanning and evaluating emerging technologies.

In summary, high process performance means satisfying the four key stakeholders-- investors, employees, customers and society-and improving the process continuously in order to guarantee long-term success.

In the next sections the approach we have used to compose PPMS will be discussed. The steps applied were the following: (1) eliciting appropriate process performance indicators; (2) determining target values for each indicator; (3) developing methods and instruments to gather the data; and (4) creating an information system that stores collected data, distributes results and provides easy access to various user categories.

Eliciting appropriate process performance indicators

In general, two approaches exist to elicit the appropriate performance indicators: using a generic set of performance indicators and picking up the right ones; or starting from scratch. The first option seems to be more efficient as it does not 'reinvent the wheel'. However, a closer examination brings some weaknesses to light. First, there is no generally accepted list of performance indicators. Propositions made are often imprecise or related to a process which is not congruent to the one being measured. Second, choosing the adequate indicators from a list requires solid and well-founded selection criteria which discriminate sufficiently. Finally, achieving a feeling of ownership and acceptance is quite difficult through this approach.

The second option-starting from scratch-appears more promising: performance indicators can be defined with an appropriate level of detail, according to the vocabulary used and precisely adapted to the process being measured. Additionally, through the task of defining appropriate indicators, personal involvement increases, and therefore identification with the result is stronger.

Having said that the 'starting from scratch' approach is preferable, a question arises: What kind of input can be used to find the appropriate performance indicators? Figure 4 shows that process performance indicators are derived either from business process goals or from the means of achieving the goals. The business process goals can be derived mainly from three sources: the enterprise-wide objectives, the business competitors and the stakeholders. These elements as well as the possible performance indicators are influenced (or restricted) by the economic, technological, social and legal environment.

The steps for identifying performance indicators

Business processes and their activities have to make a contribution to the enterprise-wide goals and the process goals, respectively. Since process improvement should be aligned with the process goals and improvement takes place where measurement is carried out, it is selfevident that process performance indicators derive from process goals, cf. Fig. 5.

Step 1: Define high-level process goals. The identification of performance indicators begins with the definition of business process goals. In order to develop more than a unilateral view it is useful to state at least one primary, high-level goal for each of the five aspects. An example of a primary goal is 'satisfied customers' (which could be seen as 'goal c 1' in Fig. 5). As such a goal is very general it has to be specified by the subsequent steps. It is important that each process goal-high-level goal as well as subgoal-be congruent with enterprisewide goals, take into account the behaviour of the competitors and, last but not least, be in line with process stakeholders' interests.
Step 2: Derive performance indicators. In order to find possible indicators for a certain goal, the following question can be asked: What is measurable and reflects the extent to which a certain goal has been fulfilled? For example, in order to measure the goal 'satisfied customers' an overall customer satisfaction index (proposed by Kristensen et al. (1992) or Kaplan and Norton (1996), p. 82) could be used; see PI (performance indicator) B in Fig. 5. However, it is not always possible to find indicators which are clearly related to the goal stated. For instance, the high-level goal 'societal responsibility' can hardly be measured directly. In this **case** , a further refinement (cf. step 3) will solve the problem.

Step 3: Derive subgoals. Since goals and their associated performance indicators may be rather general (especially during the first iteration), it is necessary to decompose them, e.g. in Fig. 5 goal c1 is decomposed into c1. 1 and c1. 2. However, a simple goal decomposition carried out by someone without specific domain knowledge is not sufficient as it does not take into account that different organizations follow different achievement strategies. Therefore, the question to be asked is: Which means or actions can be taken by the organization to fulfil a certain goal? The answer normally received has the form of a subgoal. For example, in order to raise overall customer satisfaction, on-time delivery is critical and can serve as a subgoal. And what if no means can be applied (within the process team) to accomplish a certain process goal? There are two choices: either the goal has to be moved to a higher organizational level or the process team has to be empowered in order to apply certain means.

Step 4: Refine and modify goal tree. Continuous measurement of selected performance indicators often leads to the effect that process actors emphasize the aspects measured at the expense of unstated or implicit goals (Austin, 1996). To anticipate this effect we have to ask whether measurement of the elicited indicator(s) produces unintended side-effects. If this is very likely, the goal tree has to be completed and performance indicators added if possible. For example, stressing the goal 'on-time delivery' could lead to increased stocks. In order to prevent such an unintended migration, an additional indicator, e.g. 'profit on working capital', should be added. (Go back to step 2 and iterate as often as needed.)

Figure 4.

There is a debate whether performance indicators should be focused on procedures (activities) or on results (output). The procedure-oriented community, e.g. the ISO 9000 community, argues that procedures (processes in a narrow sense) determine the result. In other words, good results are achieved through good procedures. The result-oriented community, on the other hand, argues that there is no 'one best way'. The best procedures have to be chosen according to the resources available. This may lead, for instance, to a situation whereby process team A chooses procedures g, h and j, while process team B selects procedures g, k and p to achieve an

equivalent result. What may be concluded? The more precisely the ideal procedures are known and the more uniform and predictable the resources are in their behaviour, the better performance measurement can be focused on procedures. In contrast, the more potential approaches, techniques and tools exist to achieve stakeholders' satisfaction the more the emphasis should be placed on results.

Requirements on process performance indicators

In the preceding section it was shown how performance indicators can be elicited. The aim of this section is to list the requirements which performance indicators should fulfil. According to Kitchenham (1996, p. 103) and Winchell (1996, p. 108), the main requirements are:

* Quantifiability: If performance indicators are not quantitative by nature, they have to be transformed. For instance, the performance indicator customer payment attitude could be transformed into number of days between 'invoice sent' and 'invoice paid'.

Figure 5.

Sensitivity: Sensitivity expresses how much the performance must change before the change can be detected. Therefore, a sensitive indicator is able to detect even minor changes in performance.

* Linearity: Linearity indicates the extent to which process performance changes are congruent with the value of a certain indicator. Or, conversely, a small change in the business process performance should lead to a small change in the value of a corresponding performance indicator, whereas an ample performance rise should also lead to strong change in the level of the performance indicator.

* Reliability: A reliable performance indicator is free of measurement errors. To illustrate, if a certain business process has to be rated through a given performance indicator by different experts, the results should not depend on the subjective evaluation of an individual.

* Efficiency: Since the measurement itself requires human, financial and physical resources it must be worth the effort from a cost/benefit point of view.

* Improvement-oriented: Performance indicators should emphasize improvement rather than conformity with instructions. Therefore, measuring billing errors, number of safety violations, data entry errors and the like do not create an atmosphere where feedback sessions are viewed in a positive, constructive light.

Although performance indicators may possess the attributes listed, this does not guarantee that the indicators will be used in order to improve process performance. A requirement of primary importance is acceptability. It is evident that process teams who are measured by certain indicators on one hand and have to improve process performance according to the level of certain indicators on the other hand, must perceive the selected indicators as a fair and accurate assessment instrument. If the indicators are not well received it will be unpromising to establish a process performance measurement system to improve long-term performance and competitiveness. To check whether process participants consider the 'given' indicators as useful or not, a questionnaire can be very helpful.

Determining target values for each indicator
Omitting the definition of target (to-be) values would lead to a PPMS without a motivational effect. The process actors would not have the chance to work towards clearly defined goals, the common understanding of an excellent process would suffer and the idea of a shared mental model could not be put into practice. Furthermore, in the absence of to-be values it would be impossible to evaluate process performance. In short, a PPMS without given target values would be a weak concept. However, a look at

PPMS's implementation shows that managers really hesitate defining to-be values. They worry about setting them too low or too high. Since performance of processes has rarely been measured in the past, little knowledge is available in this area and it is mostly unknown what the so-called 'best of breed' process performance would be. From this point of view, the managers' behaviour is partially understandable.

What sources can be used to set realistic but challenging target values? Possible input may come from: scanning the market; asking stakeholders; competitive benchmarking; simulation and experiments; or research institutions.

Developing methods to gather data

Once indicators are identified and target values set, data sources, methods and instruments have to be defined in order to gather the necessary data. According to Sinclair (1995), the methods for gathering data can be divided into three categories: observational methods (e.g. videorecording or thinking aloud); database methods (e.g. study system records); and subjective methods, such as questionnaire and interviews. While observational methods have in common that some degree of formal 'objective' measurement is involved, database methods may offer a more precise view. However, database methods are not free of weaknesses. A major problem is that information is not collected primarily for process performance measurement and therefore important data are not available. The third class of methods, the subjective methods, has to deal with a rather fundamental problem. Sinclair describes it as follows: "The common thread here is to use people involved in the system that you wish to study as a measurement instrument... you rely on people to come to some sort of conclusion about the system, then access that conclusion as a measurement of the system" (Sinclair, 1995, p. 72). Despite this fundamental critique, it will be hard to disclaim subjective methods such as ranking methods, rating methods, questionnaires, interviews, or checklists. The main reason lies in the ability to acquire data that cannot be collected by observational or database methods. Since performance indicators belong to various dimensions, are financial or nonfinancial in nature, may be accessed internally or externally, are tangible or intangible, etc. multiple data collection methods have to be used to get a sound view of the current level of process performance.

Traditional measurement systems were based on cost accounting techniques and performance measures were derived from cost accounting information. Since financial data are usually stored in databases, one could assume that financial aspects can be measured precisely. However, a closer examination shows that most of the data available are not process-oriented, i.e. they are not tailored to assess the financial performance of a business process. Much has been written about process-based accounting (e.g. ABC), but the penetration of such systems is still very low.

Employee-related aspects are usually measured by subjective methods, such as questionnaires, or by observational methods. While traditional questionnaires were paper-based, today's IT offers the possibility of undertaking large surveys electronically and therefore much more efficiently. Other approaches not only automate conventional procedures, they try to exploit the full potential of modern IT. Such systems are known as Software Monitors (Teubner & Vaske, 1988) or as Computerized Performance and Control Systems (Grant & Higgins, 1996). The functionality of such systems ranges from simple monitors that count transactions to sophisticated software that tracks performance on a variety of measures.
An interesting point is raised by Grant and Higgins (1996, p. 222) regarding the frequency of data collection and feedback. Employees are most likely to believe that feedback is accurate if it is provided frequently. On the other hand, if data collection and feedback takes place often, the sense of being controlled is also increased.

Overall, the methods and instruments of collecting data in a PPMS are not basically new. The innovation is that various instruments are used

concurrently and focused on business processes. As data gathering takes place continuously or periodically, IT support is central.

Creating an information system that manages collected data

Information technology does not just support the process of data gathering, it is indispensable for data management. Once the current level of performance is measured for each indicator, these values have to be compared against target values and historical values, and trends have to be calculated. .Information regarding the performance gap (whether it is widening or narrowing) has to be acquired. Furthermore, it is essential in disseminating the results to the process participants, i.e. to the process manager and his/her colleagues, and to upper management. Additionally, a computerized system may ease access to the results in an ad-hoc manner for every process participant or person who is entitled to see them. Last but not least, computerized information systems facilitate data archiving and therefore support the calculation of time-series. In combination with statistical methods, cause-effect relationships may be established which are useful in assessing the impact of an organizational or technical measure on process performance.

The question may be asked "Is it useful to establish a new, additional information system in order to measure and control business process performance?" Since more and more companies are implementing so-called enterprise resource planning (ERP) systems which store operational data as well as management-related data for almost every business area, an additional information system seems to be a waste of resources. However, there are two reasons why ERP systems are rarely adequate for this task. First, ERP systems are generic systems. Thus, they are extremely large and complex (SAP's R/3" contains several thousands of tables). Modifying such systems would be very time-consuming and costly. Second, these systems are dedicated to collecting and storing financial and time-related data; in contrast, they are much less appropriate for other data categories such as performance data regarding innovation or customer satisfaction. Third, ERP systems consist of various modules which are tightly linked-local modifications are almost impossible.
All in all, a PPMS should be conceptualized as a modular, separate information system which is loosely coupled to other information systems throughout the organization. This provides some guarantee that the system can cope with the dynamic nature of business processes and their environment and with constant changes in informational needs, and that it will be able to benefit from modern technologies.

Lessons learned

As indicated earlier, the framework presented here has been applied in four enterprises. In order to set the scene, some contextual information is given. The four participating enterprises can be characterized as follows: one is a multinational pharmaceutical company; the PPMS built was destined for various support processes within the business area 'finance and accounting'. The second enterprise is a medium-sized commercial bank; the process to be supported was called 'managing mortgage applications'. The third enterprise is a small wholesaler which operates mainly in the domestic market. The fourth enterprise produces electronic components and has manufacturing sites in four different countries; the PPMS has to support the entire ordering process-it starts with 'order entry' and ends with 'checking customer's payment'. The size of the four enterprises involved varies from 50 to 60 000 employees.
Based on a cooperation with these enterprises over a period of 2 years, some conclusions can be drawn. First, we shall look at conclusions related to the elicitation of the performance indicators:

Traditional performance measurement systems were focused primarily on financial aspects, and secondarily on time-related aspects, so it is not surprising that practitioners put these two aspects in the foreground. In order to extend the discussion towards the aspects mentioned, a list which consists of examples of goals and indicators for each of the five areas can

be applied (see Kueng, 1998, p. 430).

* Process managers were induced to define such performance indicators where data needed are easily available. Thus, it is important to state the process goals clearly and to verify the indicators selected by several quality criteria.

* The construction of a sound set of process goals and performance actors is a creative and time-consuming step. It may take twice as long as the composition of the technical part. However, the intensive discussions about the aim and object of business processes will lead to a deeper, more customer-oriented understanding. The discussions raised a lot of questions such as "Why do we carry out the activity in this way if our customer needs that?" We believe that a solid agreement on the goals and outcomes of a business process is an essential step towards establishing a performance-based process management.

* Highly detailed and decomposed performance indicators (low-level metrics) are usually regarded as valuable as they give very specific information. However, there is a price to pay: as soon as a business process is slightly modified, these indicators are out of context and can no longer serve as a guide for performance improvement.

Second, some more general conclusions are listed:

The role of a business process manager, or process owner, is essential. He or she has to play the role of a leader, an entrepreneur and a negotiator. If his/her competency is severely limited and decision-making power is restricted, it will be very time-consuming to implement a PPMS and the PPMS cannot be deployed effectively as the necessary changes cannot be realized by the process team.

A PPMS is designed and implemented with substantial involvement on the part of senior management. For a project such as the implementation of a PPMS the endorsement and personal support of senior management must be given. Commitment of senior management without empowered process managers will not lead to success. Cross-departmental communication (e.g. human resources department with process managers) and a cooperative approach are absolutely necessary. Even then, it is very difficult to institutionalize process performance management into management thinking and into the daily operating practices of an organization.

Finally, some conclusions related to the use of a PPMS:

* Data collection must be made as easy as possible. The more effort process actors must put into non-original activities the more difficult it is to convince them that performance measurement is essential. One approach is 'automating data collection'; even more effective is 'minimizing the amount of required data'.

* One of the crucial requirements for effective use of a PPMS is the acceptance of the chosen indicators. Hence, it is essential to ensure that process participants can express themselves as to whether they consider the 'given' goals and indicators useful or not. The application of a questionnaire (in a first round) and face-to-face communication (in a second round) have proved useful. Based on an empirical study, Sinclair and Zairi came to the conclusion that personal involvement was vital for effective performance measurement (Sinclair & Zairi, 1996, p. 378).

* Measurement dysfunctions (aspects measured are improved at the expense of aspects not measured, though they may be relevant) exist and cannot be fully excluded--even if one tries to implement a balanced set of indicators. In order to anticipate an unintended shift, a more fruitful but demanding approach may be for the process team to establish a common understanding, an understanding where goals, objectives and values are shared as far as possible among the individuals.

* PPMSs in themselves do not have the ability to improve competitiveness per se. If, however, PPMS are used in conjunction with social transformation such as team orientation, a changed attitude towards openness, etc. the potential seems to be significant.

Summary

Several years ago, the business process community was debating whether the performance of business processes should be increased by a radical approach (BPR) or stepwise (continuous improvement CI). Now there is some sort of agreement that both approaches have the right to exist and some companies do combine the two approaches successfully. From Fig. 6 we can infer that the dual approach is shared by us.

First, based on process goals, a process model has to be composed (this activity may take place within a BPR project) or modified (as part of a CI program). In the next step, the business process has to be implemented (BPR) or modified (CI). Then a number of business cases (instances of a business process) are carried out. The next step, measurement of process performance-based on both indicators and target values-is usually not part of a BPR project. This is clearly a shortcoming. In a real CI program, on the other hand, performance measurement is an integral part. The same is true for the two other steps, 'feed back information' and 'define actions'. However, methods and tools to support process performance measurement and feeding relevant information back are still rare. A PPMS should alleviate this shortcoming.

To summarize the approach described in this paper, we believe that process performance measurement is a necessity for a modern process-oriented organization. Based on our experience with several enterprises, it seems very unlikely that a universal set of performance indicators can be applied successfully to all business processes. Thus, performance indicators must be process-specific and have to be derived from both the strategic enterprise-wide goals and the process goals. Performance measurement, by itself, does not show which actions are to be taken in order to improve a process. However, it is a means to give the process a clear direction, to identify areas of weakness, to evaluate process performance comprehensively and to assess the impact of previous process changes.

Figure 6.

References

AuSTIN, R. (1996) Measuring and Managing Performance in Organizations (New York, Dorset House Publishing).

ANDERsoN, R. (1996) Preventive activities at the workplace in the European Union. In: 0. BROWN & H.

HENDRICK (Eds) Human Factors in Organizational Design and Management V (Amsterdam, North-Holland), pp.327-332,

BASILI, V. & RoMBACH, D. (1988) The **TAME** project: towards improvement-oriented software environments, IEEE Transactions on Software Engineering, 14, pp. 758-773.

BERRY, L. & PARAsuRAmAN, A. (1997) Listening to the customer: the concept of a service-quality information system, Sloan Management Review, 38, pp. 65-76.

BiERi, B. (1995) Kybernetisches Produktions-Controlling mit Hilfe von Kennzahlen, Dissertation, University of St Gallen.

BRANNiCK, M. & PRINCE, C. (1997) An overview of team performance measurement. In: M. BRANNicK, E. SALAS & C. PRINCE (Eds) Team Performance Assessment and Measurement: Theory, Methods, and Applications (Mahwah, NJ, Lawrence Erlbaum Associates), pp. 3-16.

BRYNJOLFssoN, E. (1993) The productivity paradox of information technology, Communications of the ACM, 36,pp.67-77.

CHANG, R. & YOUNG, P. (1996) Measuring Organizational Improvement Impact (London, Kogan Page). COOPER, R. & KAPLAN, R. (1991) Profit priorities from activity-based costing, Harvard Business Review, 69, pp.130-135.

COOPER, R. & KAPLAN, R. (1998) The promise-and peril-of integrated cost systems, Harvard Business Review, 76, pp. 109-119.
ECCLES, R. (1991) The performance measurement manifesto, Harvard Business Review, 69, pp. 131-138. EFQM (1999) The EFQM Model 1999; available: http://www/efqm/org/, accessed 15 April 1999.

GILLIEs, A. (1997) Software Quality: Theory and Management, 2nd Edn (London, Thomson Computer Press). GLAD, E. & BECKER, H. (1996) Activity-based Costing and Management (New York, Wiley).

GRANT, R. & HIGGINS, C. (1996) Computerized performance monitors as multidimensional systems: derivation and application, ACM Transaction on Information Systems, 14, pp. 212-235.

HAKES, C. (1996) The Corporate Self Assessment Handbook, 3rd Edo (London, Chapman & Hall).

HAMMER, M. & CHAMPY, J. (1993) Reengineering the Corporation: A Manifesto for Business Revolution (New York, Harper Business).

HARRINGTON, J. (1991) Business Process Improvement. The Breakthrough Strategy for Total Quality, Productivity, and Competitiveness (New York, McGraw-Hill).
ISO (1994) International Standard ISO 8402, Quality Management and Quality Assurance: Vocabulary (Geneva, International Organization for Standardization).

JONES, T. (1995) Instrumental stakeholder theory: a synthesis of ethics and economics, Academy of Management Review, 20, pp. 404-437.

JURAN, J. & GRYNA, F. (1993) Quality Planning and Analysis: From Product Development through Use (New York, McGraw-Hill).

KAPLAN, R. & NORTON, D. (1996) The Balanced Scorecard: Translating Strategy into Action (Boston, MA, Harvard Business School Press).

KITCHENHAm, B. (1996) Software Metrics: Measurement for Software Process Improvement (Cambridge, MA, Blackwell).

KRISTENSEN, K., KANji, G. & DAHLGAARD, J. (1992) On measurement of customer satisfaction, Total Quality Management, 3, pp. 123-128.

KUENG, P. (1998) Supporting BPR through a process performance measurement system. In: P. BANERJEE, R. HACKNEY, G. DHILLON & R. JAIN (Eds) Business Information Technology Management (New Delhi, HarAnand Publications), pp. 422-434.

KUENG, P. & KAWALEK, P. (1997) Goal-based business process models: creation and evaluation, Business Process Management journal, 3, pp. 17-38.

MBNQA (1999) Malcolm Baldrige National Quality Award Criteria 1999; available: http://www.quality. nist.gov/, accessed 15 April 1999.

McLELLAN, M. (1996) Workflow metrics: one of the great benefits of workflow. In: H. OSTERLE & P. VOGLER (Eds) Praxis des Workflow-Management (Braunschweig, Vieweg Verlag), pp. 301-318.

MENDELow, A. (1983) Information systems for organizational effectiveness: the use of the stakeholder approach, Proceedings of the IFIP WG8.2 Working Conference on Beyond Productivity, Minneapolis, 22-24 August.

MOONEY, J., GURBAXANI, V. & KRAEMER, K. (1996) A process oriented framework for assessing the business value of information technology, The DATA BASE for Advances in Information Systems, 27, pp. 68-81.

PAULK, M. (1995) The Capability Maturity Model: Guidelines for Improving the Software Process (Reading, MA, Addison-Wesley).
PARASURAMAN, A., BERRY, L. & ZEITHAML, V. (1988) SERVQUAL: a multiple-item scale for measuring consumer perceptions of service quality, Journal of Retailing, 64, pp. 12-40.

PARASURAMAN, A., ZEITHAML, V. & BERRY, L. (1994) Reassessment of expectations as a comparison standard in measuring service quality, Journal of Marketing, 58, pp. 11 1- 124.

PRESTON, L. & SAPIENzA, H. (1990) Stakeholder management and corporate performance, The Journal of Behavioral Economics, 19, pp. 361-375.

ROSE, K. (1995) A performance measurement model, Quality Progress, February, pp. 63-66.

SEI (1998) SW- **CMM** v2.0, Draft C; available: http://www. **sei** .cmu.edu/activities/ **cmm** /, accessed 15 April 1999.

SINCLAIR, M. (1995) Subjective assessment. In: J. WILDSON & N. CORLETT (Eds) Evaluation of Human Work: A Practical Ergonomics Methodology, 2nd Edn (London, Taylor & Francis), pp. 69- 100.

SINCLAIR, D. & ZAIRI, M. (1996) Assessing the effectiveness of performance measurement systems: a **case** study, Total Quality Management, 7, pp. 367-378.

SPECTOR, P. (1997) Job Satisfaction: Application, Assessment, Causes, and Consequences (London, Sage Publications).

ST. CLAIR, G. (1997) Total Quality Management in Information Services (London, Bowker-Saur).

TEUBNER, A. & VASKE, J. (1988) Monitoring computer users' behaviour in office environments, Behaviour and Information Technology, 7, pp. 67-78.

WFMC (1996) The Workflow Management Coalition Specification: Terminology & Glossary. Document Number WFMC-TC-101 1, Document Status: Issue 3.0, February 1999, available: http://www/aiim.org/ wfmc/standards/docs.htm, accessed 7 September 1999.

WINCHELL, W. (1996) Inspection and Measurement in Manufacturing (Dearborn, MI, Society of Manufacturing Engineers).

ZAIRI, M. (1996) TQM-based Performance Measurement: Practical Guidelines (Cheltenham, Stanley Thornes).
PETER KUENG

University of Fribourg, Institute of Informatics, Rue Faucigny 2, 1700 Fribourg, Switzerland

Correspondence: Peter Kueng, University of Fribourg, Institute of Informatics, Rue Faucigny 2, 1700 Fribourg, Switzerland. Tel: + 41 26 300 8335; Fax: + 41 26 300 9726; E-mail: peter. kueng@unifr. ch

DESCRIPTORS: Competitive advantage; Business process reengineering; Measurement; Performance evaluation; Studies
CLASSIFICATION CODES: 5320 (CN=Quality control); 9130 (CN=Experimental/Theoretical)
PRINT MEDIA ID: 34507

5/9/3 (Item 1 from file: 148)
DIALOG(R)File 148:Gale Group Trade & Industry DB

05910516 SUPPLIER NUMBER: 12503495 (THIS IS THE FULL TEXT)

CASE : a testbed for modeling, measurement and management. (Special Section: Computer-Aided Software Engineering in the '90s)

Tate, Graham; Verner, June; Jeffery, Ross
Communications, v29, n4, p65(8)
April, 1992
ISSN: 0010-356X LANGUAGE: ENGLISH RECORD TYPE: FULLTEXT
WORD COUNT: 5635 LINE COUNT: 00459

TEXT:

Human activity in the developed world strives not only to maintain status quo activities and lifestyles, but to improve on them. In particular, the application of technology has been focused on improvement. Technology is central to organized society's efforts to improve the lot of individuals and organizations, regardless of one's opinions of the success or failure of instances of technological application or of the ultimate nature of improvement.

This ethos of improvement or "doing better" has strongly influenced attitudes toward software development and maintenance. From the software crisis of the mid-1960s, well described in |6
, to the present day, many concepts, methodologies, languages, tools and techniques have been introduced with the aim of improving the software process and its products. Particular initiatives which we will examine here are **CASE** and the software improvement paradigms of the Software Engineering Institute (**SEI**), Software Process Capability Maturity Model (**CMM**), |10, 13, 14
and the University of Maryland's Tailoring a Measurement Environment (**TAME**) project |1, 2
.

CASE aims at greater automation of software production. Just as CAD/CAM has brought integrated design tools to the engineering of physical systems, **CASE** is bringing analogous tools to the more abstract engineering of software. Ultimately, the motivation for tool use is economic--for competitive advantage. There are many aspects to competitive advantage, including time-to-market, productivity, quality, product differentiation, distribution and support. Software engineering, however, has a narrower scope, comprising software definition, design, production, and maintenance. **CASE** aims to improve these activities through the use and integration of software tools.

Software improvement has recently received more explicit emphasis, together with a firmer conceptual and empirical basis, through the work of the **SEI** on the **CMM** |10, 13, 14
and the work of Basili and Rombach on the **TAME** project |1, 2
. Central to both of these major research efforts has been the characterization and improvement of the software process. There are differences in the two improvement paradigms, which will be examined briefly later, but they agree in assigning a central role to software metrics for both software process characterization and improvement.

The term software process has been used to designate the complex process by which software is developed, from conception to operation. Traditionally, the software process has been depicted informally in software development life cycle (SDLC) diagrams, usually augmented by textual descriptions. Recently, however, more formal approaches to software process modeling have emerged |12, 18
with the construction of more precise and detailed models. The computer-aided enaction of such models by developers can potentially provide a degree of definition and traceability which has been difficult to achieve in the past and should lead to the collection of much better data about software development.

The important developments in **CASE** , software maturity and improvement, software metrics, software process modeling and enaction, are brought together in this article and their relationships explored. The

salient relationships between contributing areas are illustrated in Figure 1. A modeling, measurement and development tool architecture is presented which integrates **CASE** , metrics, and process model enaction and thereby provides a framework and testbed for the understanding, management, and improvement of the software process.

CASE , Software Improvement and Metrics

In view of the software improvement intent of **CASE** , it is appropriate to examine **CASE** in relation to major software improvement models, namely the **CMM** and the **TAME** model.

CASE and the **SEI** Capability Maturity Model

The **CMM** is a five-level model in which higher levels indicate greater software process maturity which is in turn associated with increasing productivity and decreasing risk. Level 1 (Initial) represents the crisis-driven, ad hoc development which is still all too common. Level 2 (Repeatable), while still intuitive and dependent on individuals, exhibits regularity in repeating previously mastered tasks and the beginnings of manageability. Level 3 (Defined) is characterized by a specified and institutionalized software process, no longer so dependent on individuals, and by having a Software Engineering Process Group to lead process improvement. At Level 4 (Managed) the process is measured and controlled in the sense that relationships between activities are understood quantitatively so that, for example, variations in early activities can be analyzed to determine their expected effects on later activities. Managers have a firm quantitative basis for their plans and decisions. Level 5 (Optimizing) provides not just for the management of a defined process using automatic data collection but for change and optimization of the process itself.

The **CMM** has been criticized on a number of counts |5

. Much of the criticism is, in fact, a tribute to its profound impact on U.S. software producers. Some authors noted that the model is strikingly indifferent to the use of technology and automation for process improvement, that organizations and projects straddle the levels, that the multihurdle grading system has statistical and methodological flaws and that the model is unproven. ("It appears unlikely that such ratings have any meaningful correlation to the actual abilities of organizations to produce high-quality software on time and within budget" |5

.) The evaluation program itself, and the importance to some organizations of getting high scores, may distort software improvement to fit the model framework. Other authors |11

have responded to these criticisms, noting that management is more important than technology, that experts have been unable to agree on the technologies necessary for software process improvement, and that the software capability evaluation instruments are evolving and subject to periodic revision. Other questions could be raised, such as the applicability of the model to application domains other than the U.S. defense domain within which it has largely been applied to date |5, 14

. It would be unrealistic to regard the model as perfect, and equally unsatisfactory not to regard the profound effect which it is having on the software community as being mainly beneficial. From our point of view, the important issues raised by the **CMM** are the importance of improving software capability, the establishment of an improvement framework, and the need for metrics. We also examine the role of **CASE** in software improvement, which **CMM** does not seem to address adequately. The relevance of specific **CASE** tools to different **CMM** maturity levels has been considered by |19

.

The introduction of **CASE** is seen by many organizations as a means of improving the consistency, repeatability and definition of their software process, as well as either the quality of their software products or the productivity of their software process, or both |17

. Since **SEI** maturity Level 2 is characterized by the label Repeatable and Level 3 as Defined, it can be argued that **CASE** would be seen by many, rightly or wrongly, as a major means of software maturity advancement to levels 2 and 3 if they were to consider their use of **CASE** in a **CMM** context. There is much work on **CASE** technology transfer |7, 15, 16

which indicates that **CASE** alone is not enough for improvement and

that appropriate motivation, education and management are also necessary within a carefully planned introduction program. Nevertheless, **CASE** is often seen as the major technological agent on which improvement is focused.

Progress up the maturity ladder also necessitates the use of appropriate metrics to assess the effects of tools, technologies, education and management and to measure aspects of the ascent from rung to rung. The **SEI CMM** implies the need for software metrics in the five maturity levels as follows |14, 20

.

* Level 1, Initial: estimating software size, estimating resource needs, developing schedules, software quality assurance, gathering data on code and test errors.

* Level 2, Repeatable: need for orderly methods for tracking, controlling and improving the quality of software or of the software process; need for action plans directed at long-term software process improvement.

* Level 3, Defined: the three key challenges are all metrics related: process measurement, process analysis and quantitative quality plans.

* Level 4, Managed: both key process activities and major product properties are measured; changes are made to the software process specifications based on results of analyzing this data.

* Level 5, Optimizing: data gathering is automated; metrics have a process or process improvement focus, or their purpose is to support a technology improvement plan.

Thus, metrics is a key factor in raising the **CMM** software maturity level of an organization.

CASE within the **TAME** Framework

In the **TAME** project at the University of Maryland, Basili and Rombach have developed a methodology for improving the software process by tailoring it to specific project goals and environments |1, 2

. The improvement methodology is a loop with five steps:

Step 1: Characterize the environment and candidate models, methods and tools for improvement.

Step 2: Establish and quantify improvement goals and check consistency between goals and improvement agents.

Step 3: Choose first appropriate models, then appropriate methods and tools.

Step 4: Transfer the chosen technology; carry out the software project; collect, validate and analyze data, providing on-line feedback throughout.

Step 5: Do post mortem analysis and recommend new improvement goals.

In **TAME** , metrics are chosen using the Goal, Question, Metric (GQM) paradigm. Improvement goals lead to questions that must be answered to determine whether the goals are being achieved. These questions, in turn, lead to metrics which are necessary to obtain quantitative answers to the questions.

In many respects **TAME** is complementary to **CMM** . From the **CMM** point of view, the **TAME** improvement methodology can be regarded as an iterative process for climbing the capability maturity ladder. From the **TAME** point of view, **CMM** is concerned with steps 1, 2 and 5 of the **TAME** improvement methodology: characterizing the environment, at least from a maturity level point of view, identifying key metrics issues and setting improvement goals (the next rung on the maturity ladder), doing post mortem analysis (determining what maturity level has been achieved).

Software Process Modeling, Measurement and Management

In order to measure and manage the software process, we need suitable models of it. It should be defined. Not necessarily in the extended sense in which Level 3 of the **CMM** is Defined, but it should be sufficiently well defined for every resource-consuming activity, and every product or component of interest, to be comprehended within the chosen model. Traditionally, SDLC diagrams and verbal commentaries have been used to describe software development, typically illustrating variants of the well-known waterfall model and recently more sophisticated models, such as the spiral model |4

. SDLC diagrams require formalizing, however, before they can be used to define software processes, rather than just describe them. In seeking an

appropriate definition vehicle, we turn to the recent research area of software process modeling.

Software process modeling has as two of its goals greater understanding and more formal description of the software process. Two significant contributions to the field have been those of Osterweil [18] and Humphrey and Kellner [12]. Osterweil's work on process programs can be summed up by the catch-phrase title of his paper: "Software Processes are Software Too" [18]. Process programming represents software development in program form, using programming languages, notations and formalisms. Such process programs, however, are not executed (since development is still far from automatic), but enacted in a symbiosis of developer, machine and process program [23]. In contrast to the essentially activity-based or functional view of process programming, Humphrey and Kellner propose a more product-based model made up of entities, such as code modules or user manuals, states, such as non-existent, undergoing development, developed, running tests, analyzing problems, passed testing, and state transitions, for example from running tests to either analyzing problems or to passed testing.

Our software process modeling goals are somewhat different than those of many software modeling researchers. We seek to measure and manage the software process. Thus the modeled process must be measurable and manageable. For these purposes, the goals of understanding and formalization are clearly helpful, but are not enough. The difference in goals imposes a number of constraints and has a number of implications for process modeling. Notations and objects must be understandable, acceptable and, if possible, natural to software managers and developers. The granularity of activity and product decomposition must not be so coarse as to bury essential detail, such as embedded rework, or so fine as to be a measurement nuisance to developers and an unacceptable overhead. Both process programming and entity process modeling are sufficiently flexible to model software products and processes at any desired level of generality or detail. They are both, however, rather formal means of representation not intended for, nor suited to, ease of communication. During enaction, a process program can present a friendly face to the developer using appropriate dialogues, menus and graphics, but this is a different matter than communicating the structure of a process model. An adaptation of systems analysis methods is more suitable for this purpose. In this approach, data flow diagrams (DFDs), or equivalent representations, are enacted. DFDs have the advantages of being widely known, easily understood, suited to the description of asynchronous person-machine systems, and covering activity breakdown levels appropriate to developers and managers. It might be argued that DFDs are themselves, like SDLCs, too informal for enaction. However, work has been done on the formalization of DFDs for their direct execution [8] which is analogous to enaction.

The Role of Process Model Enaction

This section examines the role of process model enaction, first from a conceptual and then from a practical point of view. Conceptually, process model enaction provides a mechanism by which process models can be used for measurement and management. A software process model can be used not only to define, but also to structure, direct and record software development by having it enacted, partly by the developer and partly by the workbench on which the model is implemented. Enaction of a suitable model has great advantages for both measurement and management. If the model is clearly understood and agreed on by developers, its enaction can provide a framework within which they can work. Their activities can be clearly identified, and resource use can be recorded and directly related, at or near the time of use, to specific model activities, products and components. A suitable software process model can provide a comprehensive framework for development, and enaction can provide traceability through the model of developer effort and its effects. In concept, the role of process model enaction is a central one and its potential advantages are considerable.

From a practical point of view, enaction experiments relevant to our purposes have been conducted by Boehm and Belz [3] and by one of the authors. Boehm and Belz constructed and enacted a

process program of the spiral model. Their observations are noteworthy, including ". . . the process programming experiment led to a revised process program better reflecting the realities of Spiral Model application. In particular, it identified the importance of early process requirements, architecture and design activities and the appropriateness of the Spiral Model risk-driven approach in guiding these activities." They identified many unresolved issues in the enaction of process programs but, more important, they demonstrated the feasibility of enacting a process program.

Work by one of the authors on the enaction of process models also confirms the feasibility of enaction with two different process models, at least for small development projects in closely controlled environments. The first of two enaction models tested is shown in Figure 2. It is a simple waterfall model, with feedback flows, depicted as a top-level DFD. In practice, most of the DFD processes would be exploded to a further level of detail. The requirements and design processes would both include validation or verification steps, for example. The enacting developer first chooses an activity; then he or she choose appropriate data flows into that activity and particularizes them to specific products and components. The activity is then enacted and if it results in product changes, output flows are generated. For the simple developments chosen, all activities and products were covered by the model, all effort was recorded by activity and product/component and traffic along all data flow paths measured. The second enaction model tested was a product-based model shown in Booch diagram form in Figure 3. In this model, the developer selects a product, a specific product component, and then an operation on that component, for example "develop" or "validate/verify." Effort is recorded by the activity package and the resulting product change by the product/component package. The time and date of development will normally indicate if it is enhancement or rework. A process program has been written for this model and its enaction walked through. Once again the model was adequate for a simple development within a controlled environment.

In contrast to the spiral model |3, 4
, the two models depicted in Figures 2 and 3 are somewhat passive data collection models rather than more proactive management models. In fact, they can be regarded as different views of the same underlying product-based software process model for which a process entity model can also be constructed. The model has similarities to ISTAR |9
in terms of its product/component focus and the work contract implied in the component/activity relationship.

CASE and Metrics

As noted earlier, metrics are an essential component of both the **SEI** and **TAME** improvement paradigms and Level 5 of **CMM** has automatic data collection as a prime characteristic |14
. In the past, the effective collection of software metrics has proven difficult and expensive. Data collection has been seen as a costly overhead and a nuisance to developers who hate filling in time sheets and do not want to be bothered with recording their activities in detail. Unpublished expert estimates of metrics collection overhead costs for different organizations and purposes have variously been 8--25% (general research), 10--20% (quality), 3--5% (progress/quality), 1.5% (progress). **CASE** , particularly integrated **CASE** (I- **CASE**) or component **CASE** (C- **CASE**), offers the potential to collect measurements of on-workbench activity and products automatically, both more accurately and in greater detail than existing, largely manual methods.

It is important to distinguish between automatically collectable data and data which, by its nature, must be collected manually. The metrics data of interest include effort (related to purpose, activity and product), product/component size and complexity, and quality data, such as defects and faults. In principle, it is a straightforward matter to measure the size and complexity of all products stored in a **CASE** repository |21, 22
once one has decided on suitable, objective metrics. The automatic collection of effort and defect data is more difficult, however. On-workbench effort can be measured in terms of session lengths (allowing for periods of in-action), keystrokes, mouse clicks and **CASE** transaction counts (such as add an entity, add a relationship, delete a data type). On-workbench effort can also be related to product changes by comparing the

before and after states of affected product components. However, there is much related off-workbench activity which, by its nature, cannot be recorded automatically. This includes thinking, discussion, meetings and other miscellaneous activities. Similarly, defect and fault data arise in many different situations, such as inspections, verification, use of intermediate products in later development, testing and operation, many of which may not be directly related to workbench activity. The collection of off-workbench data is an important issue which is considered in the next section.

Suitably instrumented I- **CASE** , or C- **CASE** including a metrics component, are ideal for automatable data collection and, indeed, open new measurement possibilities |21

. Much of the required data, including product/component size and complexity, can be collected by a suitable metrics component which has read access to the **CASE** repository. However, keystroke, mouse click and transaction counts are best collected by **CASE** tools themselves. There has been some discussion, particularly at the **CASE** '89 Workshop in London, about whether or not **CASE** vendors should include metrics in their products. Many users at the workshop felt they should. Vendors, however, pointed out that users were not agreed about precisely what metrics they wanted.

CASE raises new measurement possibilities and requirements. For example, in the early stages of **CASE** development much use is made of diagrams for system modeling and design, such as DFDs, data models and structure charts. Automatic measurement of the size and complexity of such diagrams, given suitable metrics, would clearly be very useful for estimating downstream **CASE** development effort. Maximizing the automation of metrics with **CASE** should help to reduce data collection and analysis costs, increase the accuracy and consistency of size and complexity data and provide comprehensive size and complexity trend data over time. The authors are of the opinion that automatic collection of software development data, where this is possible, should not wait until Level 5 of the **CMM** but, in view of the importance of metrics to all **CMM** levels above the first, should start as early as practicable.

CASE as a Testbed for Modeling, Measurement and Management

We now need to put all the pieces into an integration model as shown in Figure 4. In this model the developer does not interact directly with the **CASE** tool set, or integrated **CASE** development environment, he or she is using. Instead, the developer enacts a suitable process model, such as that of Figure 2 or Figure 3. It is simplest conceptually to consider enaction as occurring in sessions in which a developer selects one component and one type of operation. Single-purpose sessions of this nature may be quite long (e.g., developing a data model); whereas others may be short and run together with other sessions (e.g., making consistent changes to both a data model and a DFD, working on each alternately). The single-purpose session is important as a measurement unit. Enaction of the software process model associates the session for measurement purposes with a specific model product/component and operation (e.g., "develop data model" or "test customer data entry/change module").

Before passing the developer on to the **CASE** environment, the software process model activates a metrics envelope. This takes a before-session snapshot of the repository, turns on a clock and initializes counts for on-workbench developer activity. The developer then enters the **CASE** development environment to which most of his or her effort will normally be devoted. At this early stage no inbuilt constraints have been implemented which would restrict the developer in such a way that he or she could only access the selected product/component and uses only the appropriate **CASE** transactions for the selected operation. However, **CASE** repository snapshot comparisons and transaction traces can be used as a check. We are relying on developer education and discipline, which is acceptable in a research environment. When the developer indicates the end of a session, the entry process to the **CASE** environment is essentially reversed as we back out through the metrics envelope to the software process model. An "after"-repository snapshot is taken, normally just of the selected product/component, and changes from "before" noted; the clock is switched off, the elapsed time and counts are passed to the software process model, and control also transferred to the software process model.

The measurements are logged to the appropriate activity before the developer begins another session and the cycle is repeated.

The process model thus records all product/component changes as well as on-workbench activity and effort. There are several possible ways of collecting off-workbench data. One approach is to present the developer with suitable menus requesting details of off-workbench activities at appropriate intervals, say once or twice a day. Another is to request details of any related off-workbench activity at the beginning of each session. Related design reasoning, or other relevant information, can also be captured, if desired. A third approach is to attempt to record all data within the software process model framework, so that no development is done, either on or off the workbench, except within the enaction of some software process model activity. If this approach is adopted, the model is always entered by the developer when starting any work on the project and exited only when that work is completely finished. In all cases, the scope of software process model activity must be clearly defined. It must be clearly understood what activities are included and what, if any, are excluded. If high-quality data is to be collected, it is important both that data be recorded when it is recent and clear in developers' minds and that it be recorded within the same environment, and using the same conventions, as that in which it will be used.

Figure 4 shows a project manager interaction with the software process model as well as a developer interaction. This is intended to allow for associated project management activities, such as making estimates, setting and modifying goals, and monitoring progress. For example, the sizes of all products, such as requirements, data models, DFDs, database schemas, action diagrams or structure diagrams, can be continually monitored, as can the complexity of structure diagrams. A trace of product size and complexity changes over time can give a good picture of progress and productivity throughout each product's life cycle, including development, rework, enhancement and maintenance. The software process models of Figures 2 and 3 do not include project management activities at this stage. That is the subject of a future study.

The integration model relates to parts of Steps 3, 4 and 5 of the **TAME** improvement methodology as follows. For Step 3: Appropriate software process models and methods can be built using DFDs, process programs, or some equivalent notation. **CASE** tool sets or environments can be chosen as appropriate tools. The integration model can be used in Step 4 to "carry out the software project; collect, validate and analyze data, providing on-line feedback throughout." Finally in Step 5, the comprehensive metrics collected can be used in the post mortem analysis. In relation to the **CMM** , the integration model can provide process definition as well as the metrics necessary to resolve improvement issues at all levels of maturity.

Conclusions

CASE is viewed by many organizations as an important agent of software improvement, but it is not sufficient on its own. When integrated with metrics and a suitable software process model, **CASE** can form the core of a testbed for modeling, measurement and management of the software process. It can do this for the following reasons: 1) For modeling, including empirical model validation and improvement, because **CASE** tools are employed within, and only within, the enaction of a suitable software process model. 2) For measurement, because the **CASE** tools are embedded in a metrics envelope which maximizes automated data collection and, in conjunction with the process model, provides immediacy for nonautomatable data collection and traceability for all measurement. 3) For management, because the integrated model can provide comprehensive progress reporting and can also provide data from which earlier and later activities and products can be related for the purposes of estimation, planning and monitoring.

The integrated model fits well with influential software improvement paradigms, such as **CMM** and **TAME** , providing a mechanism or platform which can meet their process definition and metrics needs. The feasibility of the integrated model has been established in a limited, controlled research environment. Much more work needs to be done, however, before it can be applied more widely in practical situations.

References

1. Basili, V.R., and Rombach, H.D. Tailoring the software process to

project goals and environments. In Proceedings of the Ninth ICSE (Monterey, Calif. Mar. 30-Apr. 2, 1987), ACM, N.Y., pp. 345-357.

2. Basili, V.R., and Rombach, H.D. The **TAME** Project: Towards improvement-oriented software environments. IEEE TSE 14, 6 (June 1988), 759-773.

3. Boehm, B.W. and Belz, F. Applying process programming to the spiral model. In Proceedings of the Fourth International Software Process Workshop, C Tully, Ed., (Moretonhampstead, Devon, UK, May 1988), pp. 11-13. Reprinted as ACM SIGSOFT Softw. Eng. Not. 14, 4 (June 1989), 46-56.

4. Boehm, B.W. A spiral model of software development and enhancement. IEEE Comput. (May 1988), 61-71.

5. Bollinger, Terry B., and McGowan, Clement A. Critical look at software capability evaluations. IEEE Softw. (July 1991), 25-41.

6. Brooks, Frederick P. Jr. The Mythical Man-Month. Addison-Wesley, Reading, Mass., 1975.

7. Corbitt, G.F., Norman, R.J., and Butler, M.C. Assessing proximity to fruition: A **case** study of phases in **CASE** technology transfer. Int. J. Softw. Eng. Knowl. Eng. 1, 2 (June 1991), 189-201.

8. Docker, T.W.G., and Tate, G. Executable data flow diagrams. In Software Engineering 86 D. Barnes and P. Brown, Eds. IEE Computing Series 6, 1986.

9. Dowson, M. ISTAR--An integrated project support environment. In Proceedings of the ACM SIGSOFT/SIGPLAN Symposium on Practical Software Development Environments (Dec. 1986), pp. 27-33.

10. Humphrey, W. Characterizing the software process: A maturity framework. IEEE Softw. (Mar. 1988), 73-79.

11. Humphrey, W.S., and Curtis, B. Comment on 'A Critical Look', IEEE Softw. (July 1991), 41-46.

12. Humphrey, W.S. and Kellner, M.I. Software process modelling: Principles of entity process models. In Proceedings of the 11th ICSE (Pittsburgh, Pa, May 1989), pp. 331-342.

13. Humphrey, W.S., Kitson, D.H., and Kasse, T.C. The state of software engineering practice: A preliminary report. In Proceedings of the 11th ICSE (Pittsburgh, Pa, May 1989), pp. 277-288.

14. Humphrey, W.S., Snyder, T.R., and Willis, R.R. Software process improvement at Hughes Aircraft. IEEE Softw. (July 1991), 11-23.

15. Merritt, P. **CASE** and culture--Observations on technology transfer. **CASE** '88 Advance Working Papers, International Workshop on **CASE** , Inc., (Cambridge, Mass. July 1988), 22: 14-16.

16. Norman, R.J., Corbitt, G.F., Butler, M.C. and McElroy, D.D. **CASE** technology transfer: A **case** study of unsuccessful change. J. Syst. Manage. (May 1989), 33-37.

17. Norman, R.J., and Nunamaker, J.F. **CASE** productivity perceptions of software engineering professionals. Commun. ACM 32, 9 (Sept. 1989), 1102-1108.

18. Osterweil, L. Software processes are software too. In Proceedings of the Ninth ICSE (Monterey, Calif., Mar. 30-Apr. 2, 1987), ACM 2-13.

19. Pfleeger, S.L. Process maturity as framework for **CASE** tool selection. Inf. Softw. Tech. 33, 9 (Nov. 1991), 611-615.

20. Pfleeger, S.L. and McGowan, C. Software metrics in the process maturity framework. J. Syst. Softw. 12 (1991), 255-261.

21. Tate, G. and Verner, J.M. Software metrics for **CASE** development. In Proceedings IEEE COMPSAC (Tokyo, Japan, Sept. 1991), 565-570.

22. Tate, G. and Verner, J.M. Approaches to measuring the size of application products with **CASE** tools. Inf. Softw. Tech. 33, 9 (Nov. 1991), 622-628.

23. Tully, C., Ed. Representing and enacting the software process. In Proceedings of the Fourth International Software Process Workshop (Moretonhampstead, Devon, UK, May 11-13, 1988), pp. 3-4.

CR Categories and Subject Descriptors: D.2 |Software Engineering : D.2.2 |Software Engineering : Tools and Techniques--computer-aided software engineering (**CASE**); D.2.8 |Software Engineering : Metrics; D.2.9 |Software Engineering : Management

General Terms: Management

Additional Key Words and Phrases: Maturity model, metrics envelope,

software capability, software maturity, **TAME**

About the Authors:

GRAHAM TATE is a professor and head of the newly established Department of Information Systems at City Polytechnic of Hong Kong. His main research interests are in information systems management and economics, software metrics and **CASE** . Author's Present Address: Department of Information Systems, City Polytechnic of Hong Kong, 83 Tat Chee Avenue, Kowloon, Hong Kong

JUNE VERNER is a senior lecturer in information systems at the University of New South Wales, Sydney, Australia. Her research interests include information systems management, software metrics and **CASE** . Author's Present Address: School of Information Systems, University of New South Wales, P.O. Box 1, Kensington, NSW 2031 Australia

ROSS JEFFERY is a professor and head of the School of Information Systems at the University of New South Wales. He has worked with many organizations in Australia and elsewhere developing cost estimation methodologies and toolsets. Author's Present Address: School of Information Systems, University of New South Wales, P.O. Box 1, Kensington, NSW 2031 Australia.

SPECIAL FEATURES: illustration; chart
INDUSTRY CODES/NAMES: TELC Telecommunications
DESCRIPTORS: Computer-aided software engineering--Usage
FILE SEGMENT: TI File 148
?

02329154 86922070

Quality initiatives in an Indian software organization: a case study

Wali, Ayoob Ahmed; Gupta, A D; Deshmukh, S G

Work Study v49n7 PP: 285-291 2000 ISSN: 0043-8022 JRNL CODE: WST
DOC TYPE: Periodical; Feature LANGUAGE: English RECORD TYPE: Fulltext
WORD COUNT: 2291

ABSTRACT: In the present economic context of liberalization and globalization, Indian organizations face many challenges. Customer focus is essential as is a corresponding emphasis on the quality of products, systems, and procedures. The Indian software industry has been recognized globally for its competitiveness built upon quality attributes such as timeliness and reliability of delivery. This paper is based on a case study carried out in one of the leading software organizations in India involved in developing a range of application software for banks, insurance companies, and financial houses. The case study work involved a survey identifying the critical success factors for TQM, and identifying how the company adopts various principles and techniques of **quality management** . The paper uses this case study to suggest a model for TQM implementation.

TEXT: Ayoob Ahmed Wali: Ayoob Ahmed Wali, is based at the Mechanical Engineering Department, Indian Institute of Technology Delhi, Hauz Khas, New Delhi, India.

A.D. Gupta: A.D. Gupta is based at the Mechanical Engineering Department, Indian Institute of Technology Delhi, Hauz Khas, New Delhi, India.

S.G. Deshmukh: S.G. Deshmukh is based at the Mechanical Engineering Department, Indian Institute of Technology Delhi, Hauz Khas, New Delhi, India.

Introduction

Today, technology has become a key driver of information systems. Although "properly" the information needs of the organization should drive the technology, with new technologies coming in faster than they can be implemented and exploited, it becomes imperative that strategic IT plans for any organization are well defined and flexible enough to exploit the competitive and commercial advantages of current and future technologies.

XYZ a software organization, offers its strategy consulting services based around enterprise modeling. A typical engagement involves the following two steps.

Mapping of the enterprise

The first step is to map the entire organization by capturing business objectives, strategies, organizational structures, operational process flows, competencies, information flows, existing IT infrastructures and other aspects of the organization in the form of an enterprise model. The model depicts the organization as a set of entities with dependencies and inter-relationships. Identification of information needs at each desk and analysis of the model developed in the first stage throws up the gaps and bottlenecks in information flows. The process also identifies the critical nature of specific pieces of information at particular desks. These needs are confirmed through discussions with representatives of the business.

Developing the information deployment strategy

Once the needs have been identified in the previous stage, the process moves on to "How". IT deployment options change almost on a daily basis, and the deployment of IS on the most appropriate IT vehicle is important. It is a wide and detailed knowledge of the options and the experience of a

variety of implementations that make the XYZ business consultants (and all such consultants) suited for assisting organizations in honing their IT strategies. Additionally, such an IT strategy consulting engagement would necessarily involve a senior business consultant from XYZ with significant experience in the business area of the client organization. A team of technical and enterprise modeling experts will support the consultant.

The implementation of software systems, and the quality factors relating to such implementation, raises different issues than for a straightforward hardware implementation. (Rook, 1986) suggests that the differences between hardware and software are:

- software has no physical appearance;

- few software quality metrics exist;

- software is much more complex than hardware;

- effects of software change propagate explosively;

- software development makes very little use of pre-existing components.

These factors impinge directly on software quality programmes and processes.

Quality management self-assessment is now of great interest to Indian companies. This is largely because of the introduction of quality award models, starting with the Malcolm Baldrige National Quality Award (MBNQA) introduced in 1987, and the European Quality Award introduced in 1991. More recently, a number of other national quality awards such as Rajiv Gandhi National Quality Award in 1991, the CII-EXIM Business Excellence Quality Award (1999), the Gold Peacock Award (1991), and the Ramkrishna Bajaj National Quality Award (1995) were introduced. These awards were based on the methodologies of the MBNQA and EQA.

These award models, the criteria and the guidelines for application are helpful in defining **quality management** in a way which enables management to more easily understand the concept (Wiele et al., 1995).

The criteria of quality awards suggest that the salient characteristics of IT organisations in terms of delivering quality products and services are:

- a focus on performance metrics - such as timely delivery;

- an ability to manage a highly knowledge-intensive work force;

- an emphasis on high performance teams;
- effective communication processes and skills;

- innovative leadership;

- a greater degree of customer contact throughout the life cycle of the project (design, development, maintenance etc.);

- appropriate motivation processes and techniques;

- an excellent interactive work environment.

Profile of the organization

XYZ is a BT-1,000 company (the top 1,000 companies in India according to Business Today magazine) with an annual turnover of Rs 10 Crores in 1998-99 (1 crore = 10,000,000 Rupees : 1$ = approx 43 Rupees). XYZ Software was founded by three enterprising professionals in 1986, when XYZ Software Workshop Pvt. Ltd was started with the objective of supporting the IT needs of Citibank, India. By 1990, it was a 200 person company and was launched as the export wing of the group. Led by a managing director, a core team was moved across and further strengthened with experienced professionals

from the industry. Though they started to make significant breakthroughs in the South East Asian region, the focus remained on the banking and financial services sector. In 1994, XYZ Software opened up a wholly owned subsidiary office in Singapore to get closer to its customers.

From consultancy and services, to the creation of innovative solutions, the transformation of XYZ Software into a "product" company, has been a deliberate process. Today, XYZ is a respected provider of customised software solutions and products for financial services, including retail banking, credit cards and leasing.

The main services offered by XYZ are:

- IT strategy consulting;

- customized software development;

- Internet/intranet solutions;

- consultancy services;

- software support and maintenance services.

The history of **quality management** in XYZ

- A brief history of **quality management** . The "quality journey" was started in 1994 aiming for ISO 9000 certification. The basic processes and documents were put in place in 1995 but certification was not achieved and the ISO 9000 programme was discontinued in mid 1996. However, the "journey" and the project were not dead altogether as a "Project summary" document was written at the beginning of 1997 to establish certain controls over the development project lifecycle. Gradually, because it was not controlled, the document (and the principles and processes within it) slowly moved from usage to non-usage. However, its use led to some serious thought about the formulation of standards and controls and led, in turn, to the formulation of the "Checkpoint Document". This Checkpoint Document was followed by all project managers (though not always totally). The document later formed the "spinal cord" of the standards set for the SEI-CMM standards (Software Engineering Institute; USA Capability Maturity Model standard).

- Performance metrics. The organization aims to deliver customer satisfaction in terms of providing high quality products, at reasonable prices, delivered on time, with effective after-sales training and services. The success of this approach is presumably reflected in the financial performance of the organization. The share price of XYZ moved from a low of 11.05 in June 1998 to a high of 40.20 in June 1999. Revenues and profits are shown in Table I.

- Quality improvement teams. In order to increase the quality of the process, SEI-CMM level processes are in place and a **software engineering process group** has been formed whose main focus is on the process improvement of the organization based on **quality management** principles.

- Quality tools. The tool used for repository purposes is Visual Source Safe. For project planning and project tracking, Microsoft Project 98 is used.

- Reward system. The "normal" company policy which encourages staff to improve and innovate within their own work/job, provides reward in the form of both stock certificates and money.

- Experience with ISO 9000. Although the process did not result in certification, it was a most useful development stage and formed the starting points for the SEI-CMM standard.

- Other quality concepts. A TQM awareness session is held every year for staff and all new joiners. Kaizen also plays an important role in the

continuous improvement process.

Market imperatives

The market strategy has always been to build on experience and to specialise in the financial services sector. Target markets were identified as South East Asia and the growing USA market.

This need to capture market share in the US was the motivating force to change the way in which the company operated, and in particular to change its attitude to quality. Project management and control was the first area to be tackled and was the first step in identifying and implementing external standards and control processes. A basic framework for project documentation was established using CheckIn and CheckOut controls and a structured repository to ensure audit trails of all projects.

These relatively tentative steps towards the establishment of control systems and standards were successful enough to lead to a re-invigoration of the concept of **quality management** and quality systems. As stated before, this coincided with the arrival or development in India of a number of internationally recognised quality award models (QAMs). Most of these models include an element of self-assessment. These enable an organisation to analyse and identify areas of weakness against the model and, hence, to initiate improvement activities. One such popular model, the CII-EXIM business excellence award, is described here.

CII-EXIM Business Excellence Award (CEBEA)

The purpose of creating this award was to set a challenge for industry to scale new heights of quality and leadership. It does this in part by creating "role model" organizations, which exemplify the application of the TQM approach to the achievement of business success - this is business excellence. The award is a collaboration between EXIM (Export Import bank), and CII (Confederation of Indian Industry) instituted and announced at the 2nd Quality Summit in 1994. The CII-EXIM Quality Award is based on the European Quality Award and is based on the following criteria:

- eadership;

- people management;

- policy and strategy;
- resources;

- processes;

- people satisfaction;

- customer satisfaction;

- impact on society;

- business results.

The concept underlying the award can be summarized as: "Customer satisfaction, people satisfaction, and a positive impact on society are achieved through leadership, which drives policy and strategy, people management, resources, and processes leading ultimately to business excellence."

The model serves as a broad structural guide for quality transformation, and the assessment categories serve as a basis for evaluating progress.

The case study methodology
The case study is based on work carried out during various field visits to XYZ software. Respondents for the survey were chosen from across the spread of departments. They completed a questionnaire and this was followed up by

both structured interviews and informal discussions. The objectives of the survey were to discover, in relation to the CII-EXIM model:

- how the participant rated the organization's current practice against each factor;

- how frequently performance is measured and ranked (to get some indication of performance);

- how senior managers practise XYZ **quality management** principles, tools and techniques.

There were seven respondents to the questionnaire. A self-assessment score (SAS) was computed to identify "quality performance".

Discussion and results

The results of the survey are shown in Table II. It is evident that for some of the factors (e.g. resources), the variability (in terms of coefficient of variation) in respondent's perception is significantly less than for others (e.g. people satisfaction).

The factors "People satisfaction" and "Impact on society" have significantly higher values for the coefficient of variance, showing perhaps the bias of the mindset of Indian management towards their employee and society.

Table III shows that the SAS for XYZ in 1999 was 718.7 (on a scale of 0 to 1,000). This was based on the collective perceptions of the managers within the company. The scoring evolved based on sessions explaining and discussing the use of the model as a self-assessment tool and asking managers both to score the organization against the identified factors and to rank the importance of factors. The relative importance of factors is shown in Table IV.

The model is based on a number of factors and the case study shows that some interaction is evident amongst these factors.

The TQM model (Figure 1) presupposes that there is some system for performance measurement in place in the organization. A prioritized list of such measures is given in Table V.
The system captures performance on a variety of dimensions (both financial as well as non-financial). As expected, customer satisfaction is the most important for XYZ. Innovation/technology and productivity follow. Interestingly, financial measures are in 4th position. The factors that make up the measures, and the frequency of measurement, are shown in Table VI.

Summary

In summary the approach followed by XYZ (and perhaps a generic model for **quality management**) is as follows:

(1) Compute a self-assessment score (SAS) based on CII-EXIM model. This involves various functional heads in brainstorming mode.

(2) Based on this SAS, identify and prioritize areas needing improvement.

(3) Devise quality initiatives (QI) to improve SAS. Involve people at all levels in implementing these QI.

(4) Measure the performance in qualitative and quantitative terms to study the impact of the QI.
Repeat steps (1) to (4) until a satisfactory SAS is obtained.

References

1. CII-EXIM Award for Business Excellence (1999), Application Brochure and Guidelines for Companies.

2. Gold Peacock Award Guidelines (1991), Institute of Directors, Delhi, India.

3. Ramakrishna Bajaj National Quality Award Guidelines (1995), Indian Merchants Chamber (IMC), Mumbai, India,

4. Rook, P. (1986), "Controlling software projects", Software Engineering Journal, Vol. 3 No. 10, pp. 108-17.

5. Wiele, Tor van der, Barrie, D., and Roger, W. (1995), "Surveying quality success - self-assessment", The Magazine of Continuous Quality Improvement, July, pp. 21-4.

Caption: Table I; Financial status of XYZ; Table II; Mean and standard deviation for critical factors; Table III; Present SAS (1999) as assessed by organization; Table IV; List of ranking performance metrics based on survey result; Table V; Quality initiative undertaken by XYZ; Table VI; Performance indicators and how frequently the organization determines them; Figure 1; Linkage between CSFs and performance indicators

GEOGRAPHIC NAMES: India

DESCRIPTORS: Studies; Software industry; Quality control; Technological planning; Self evaluation
CLASSIFICATION CODES: 9179 (CN=Asia & the Pacific); 9130 (CN=Experimental/Theoretical); 8302 (CN=Software and computer services); 5320 (CN=Quality control)
PRINT MEDIA ID: 118

13/9/4 (Item 4 from file: 15)
DIALOG(R)File 15:ABI/Inform(R)

01430801 00-81788
Combining quality and software improvement
Hollenbach, Craig; Young, Ralph; Pflugrad, Al; Smith, Doug
Communications of the ACM v40n6 PP: 41-45 Jun 1997 ISSN: 0001-0782
JRNL CODE: ACM
DOC TYPE: Journal article LANGUAGE: English LENGTH: 5 Pages
SPECIAL FEATURE: Charts Diagrams References
WORD COUNT: 2007

ABSTRACT: PRC, a subsidiary of Litton Industries, is a systems and software integrator whose primary customers are national government agencies. PRC based its software process improvement (SPI) on an integration of its corporate quality improvement (QI) program and the model-based initiatives of the Software Engineering Institute (SEI). Currently, over 75% of PRC's senior management team are formally trained in QI, with over 100 process management systems in place company-wide. Quality teams use a 7-step problem solving process called the QI story. Quality in daily work identifies, controls, and improves key work processes. Priority management focuses on achieving breakthrough improvements in the highest priority areas. The program and its results are described in detail.
TEXT: Selecting a process improvement paradigm that fits the culture of an organization can provide dramatic results. This article describes how PRC based its software process improvement (SPI) program on an integration of its corporate quality improvement (QI) program and the model-based initiatives of the Software Engineering Institute (SEI). The commitment to QI throughout PRC nurtures a culture characterized by rapid problem-solving, strong senior management support, a common process definition language, visible process improvement, and a network of processrelated teams. PRC's SPI has leveraged this QI culture to mature rapidly to Level 3 and to establish a foundation for Levels 4 and 5 of the Capability Maturity Model (CMM) for software.

PRC, a subsidiary of Litton Industries, is a systems and software integrator whose primary customers are national government agencies. PRC has 5,600 employees in criminal justice/public safety, defense, document imaging, education/training, electronic commerce, global command and control, health/medical, intelligence, legacy systems, transportation, and weather/energy/space markets. PRC maintains four core competencies: open systems, program management, software engineering, and multimedia/imaging. The SPI program spans all PRC business units and worldwide locations. Prior to 1993, PRC approached SPI in the traditional fashion-a group within our Technology Center was responsible for "getting PRC to Level 3." This group performed assessments on two pilot projects, designed corporate processes, and wrote manuals that defined software engineering and SPI practices; yet no significant institutionalization beyond the pilot projects occurred.

In 1991, PRC investigated Total **Quality Management** (TQM) programs, selected a TQM vendor, and initiated pilot programs. This manufacturingbased program failed to take hold in PRC's services environment. Under strong new senior leadership, PRC tried again, this time using the QI process developed by Florida Power & Light (FPL). At that time, FPL was the only U.S. company to win the Deming Award for Quality. This approach is supported by a FPL spin-off-Qualtec Quality Services-and has three basic components: quality teams, quality in daily work (QIDW), and priority management. Currently, over 75% of PRC's senior management team are formally trained in QI, with over 100 process management (QIDW) systems in place company-wide. **Quality management** boards (QMB), consisting of managers and their direct reports, oversee the implementation of QI at corporate and business unit levels. The QMB initiates and acts as a steering committee for QI teams.

Quality teams use a 7-step problem solving process called the "QI story."

Many of these teams cross functional areas. A QI lead team follows particular QI teams in the execution of their QI story or in the definition of processes, and facilitates the teams' recommended problem solutions.

Quality in daily work (QIDW) identifies, controls, and improves key work processes. We first identify our top-priority work processes and determine which need improvement. For those, we identify customers, determine what satisfies our customers (their valid requirements), and document the process to meet requirements using flowcharts and text. Next, we add targets and indicators to measure the process and the quality of its outcome. Using these indicators according to statistical process control principles, we monitor and modify our processes until they are stable and capable of meeting customers' needs. Then, we standardize and replicate them elsewhere in the business.

Priority management focuses on achieving breakthrough improvements in the highest priority areas. PRC's executive team conducts "voice of the customer" analysis to determine what is most important to customers, and "voice of the business" analysis to determine what is most important to employees and other key stakeholders.

Applying QI to SPI

In early 1993, the U.S. Air Force released an RFP requiring a software capability evaluation. As a result, PRC senior management created the Phoenix team in March 1993 as a cross-project quality improvement team chartered to apply QI to SPI, to perform CMM-based assessments on 10 major software projects, and to produce a SPI plan to effect improvements at project and corporate levels. The team was to exist regardless of the outcome of the RFP (which was eventually retracted). Meanwhile, PRC could see the Phoenix team mechanism was working.

Phoenix team projects represented a range of size and a variety of clients and application domains, but shared a common element: a desire to improve their software processes. Team representatives were those who were "so key to their projects that they could not be spared;" they became local SPI champions.

Each project initiated two QI teams: a QMB and a **Software Engineering Process Group** (SEPG). The SEPGs are QI lead teams and report to their QMBs. This ensures that SEPGs are integrated into our continuous improvement program. As lead teams, SEPGs and QMBs initiate QI teams. Supporting the SPI program at the corporate level are several full-time staff positions-including the Technology Center SEPG, which derives its tasking from the PRC SEPG and the Director of the Software Engineering Core Competency. The combination of line personnel involved in SPI and Technology Center full-time SPI resources has been critical to PRC's success.

The Phoenix team used the QI story as its methodology, but it supplemented freely with the tools and techniques provided by the SEI. The team found the two methodologies complemented each other; the SEI CMM methodology had a strong and well-established model that provides specific criteria, targets, and measurements, whereas the QI story provided a thorough problem-solving technique and set of tools. Table 1 shows where the SEI tools and techniques were inserted into the QI story.

In 1994, PRC initiated Phoenix II, beginning improvements on another set of 12 projects. Representatives from these projects learn QI and SPI tools and techniques, assess their projects using the PRC maturity questionnaire, build software process improvement plans, encourage each other, and broadcast their results within the company. A Phoenix III team began in September 1996; and PRC plans additional Phoenix teams in the near future to coordinate selected SPI efforts within other Litton divisions.

Measuring Results

We performed a second set of assessments in January 1994, and have perform them annually ever since. We analyze the findings from the assessments to focus our plans each year (see Figure 1). Project and company objectives and priorities are incorporated into the SPI plans so they maintain a business case for SPI while countering weaknesses from the assessment, institutionalizing strengths, and progressing toward higher maturity levels. Since our goal is not just project but organizational maturity, we continue to reexecute the QI story each year. Our assessments have continued to show marked improvement in process maturity, which translates to achievements like a reduction in critical defect density of 49.9% between releases, and a time-to-market reduction of 28.6%.

A case study reporting the experience of 11 software projects in reusing 120 processes from 1991-1995 indicated the time to develop a project-specific process was dramatically lower through reuse, showing up to a 10 to 1 decrease in time to tailor a process [3].

In July 1996, PRC's Systems Integration business unit contracted an outside firm to conduct a Software Capability Evaluation, an independent assessment of PRC's progress in achieving CMM maturity. The firm certified the business unit and the six evaluated projects at CMM Level 3. It took them 39 months to move from Level 1 to Level 3, well below the mean of 55 months [1].

Costs and Lessons Learned

The cost of SPI is shared within PRC as an overhead cost by the four business units, which invest in SPI based upon their organizational SPI commitments, goals, and customer needs. PRC has been engaged in QI/SPI for four years, spending about $1 million a year, an average of $470 per software engineer annually. Table 2 compares PRC SPI cost data with those from 13 organizations representing a variety of maturity levels [2]. Top management commitment to both QI and SPI has been essential to achieving accelerated maturity. The CEO enforces and rewards SPI involvement, advocates the program at the PRC QMB, and hosts a semiannual Executive Sponsor Status Review. SEPG representatives, program managers, and senior managers, up to and including the CEO, attend the status review, where results and issues are shared company-wide, project success stories are showcased, and every project displays their "CMM poster," detailing current status and progress. To meet middle managers' need for planning information, a "Managing Software Process Improvement" course is offered.

(Table Omitted)

Captioned as: Table I. QI and SPI tools and techniques used in the quality improvement story

The biggest challenge to continued progress in SPI is maintaining momentum and visibility. Key countermeasures are the status review, lead team briefs, PRC-wide SPI symposia, internal technical articles, capability evaluation readouts, SEI visits, and wide participation by line organizations.

Top down organizational goals drive business unit objectives; SPI goals are included in personal objectives of key management. We consider QI and SPI as the "way we do business." It is not optional. This has required a substantial cultural change both for PRC and, in some cases, for our customers.

We have found that the things that are measured are the ones that improve, but PRC experienced early difficulty in institutionalizing measurement. Keys to our current success are:

*Active management involvement

*Trained project champions

*A full-time metrics advocacy function *Higher project maturity, which motivates measurement

(Chart Omitted)

Captioned as: Figure I. Results of PRC CMM-based assessments from 1993 to 1996

(Table Omitted)

Captioned as: Table 2. Comparison of PRC and industry SPI costs

Training courses were developed for PRC corporate processes, including how to tailor them for specific projects. A cadre of SPI instructors is maintained to teach the "PRC way." As individuals move among projects, the basic processes, proceduress methods, and tools are familiar, and are the product of continuous improvement.

In 1994, the PRC SEPG converted its electronic Process Asset Library, a collection of SPI-related assets, to web-based technologies. The number of assets now totals over 1,000, including the corporate SPI processes, training materials, and related artifacts. When project "Best of Breed" processes failed to take hold, we used domain engineering techniques to build reusable processes.
The PRC Phoenix SPI Reference Manual provides information to every software engineer about PRC's software engineering policies, the CMM, the asset library, PRC's SPI Training Program, and other PRC-developed SPI tools. These tools include a CMM poster that displays a project's maturity, the Process Asset Library, and the PRC Maturity Questionnaire, which automates analysis and reporting of an expanded SEI questionnaire.

Plans for the Future

In the near term, PRC plans to replicate our Phoenix process, reaching more projects and sites. We also expect to support SPI for small projects, expand our use of measurement, assess the personal software process, and support and extend our web-based asset library. Moreover, we will establish Level 4/5 "potential maturity" through process definition, documentation, and training of all Level 4 and 5 key process areas. Each of these plans relies on elements of the QI infrastructure. Of particular value to certain Level 4/5 processes (for example, quantitative process management, process change management, defect prevention, and technology change management) is the requirement embedded in our QI/QIDW system for quality and process indicators. Therefore, improved measurement activities and use of statistical control techniques will result in additional maturity gains. The synergy between PRC's QI and SPI programs enables us to plan aggressively for Level 4 and 5 maturity at selected divisions, sites, and projects.

Reference:

REFERENCES

1. Hayes, W. and Zubrow, D. Moving On Up: Data and Experience Doing CMM-Based Process Improvement. CMU/SEI-95-TR-008, Soft. Eng. Inst., Aug. 1995.

2. Herbsleb, J., et al. Benefits of CMM-Based Software Process Improvement: Initial Results. CMU/SEI-94-TR-13. Soft. Eng. Inst., Aug. 1994
3.Hollenbach, C., and Frakes, W. Software Process Reuse in an Industrial Setting. In Proceedings of the Fourth International Conference on Software Reuse. M. Sitaraman, Ed., IEEE Comp. Soc. Press, New York, April 1996.

Author Affiliation:

CRAIG HOLLENBACH (hollenbach_craig@prc.com) is a staff engineer at PRC, Inc., in McLean, Va. RALPH YOUNG (young_ralph@prc.com) is the director of systems engineering as well as systems and process engineering at PRC, Inc., in McLean, Va.

AL PFLUGRAD (pflugrad_al@prc.com) is a senior manager at PRC, Inc., in McLean, Va.

DOUG SMITH (smith_doug@prc.com) is a principle engineer at PRC, Inc., in McLean, Va.

COMPANY NAMES:
PRC Inc
GEOGRAPHIC NAMES: US

DESCRIPTORS: Quality control; Software industry; Strategic management; Case studies
CLASSIFICATION CODES: 9190 (CN=United States); 5320 (CN=Quality control); 5240 (CN=Software & systems); 8302 (CN=Software and computer services); 2310 (CN=Planning); 9110 (CN=Company specific)
?

17/9/4 (Item 4 from file: 15)
DIALOG(R)File 15:ABI/Inform(R)

00759574 94-08966
Measurement: The key to application development quality
Walrad, Charlene; Moss, Eric
IBM Systems Journal v32n3 PP: 445-460 1993 CODEN: IBMSA7 ISSN: 0018-8670 JRNL CODE: ISY
DOC TYPE: Journal article LANGUAGE: English LENGTH: 16 Pages
SPECIAL FEATURE: Charts Diagrams References
WORD COUNT: 7784

ABSTRACT: Application development quality and productivity have been identified as being among the top 10 concerns of information systems (I/S) executives in both 1991 and 1992. In exploring the role of measurement in the pursuit of I/S application development quality and productivity, it is noted that both business value measures and I/S process and product measures are required to satisfy the information needs of I/S management. In describing the relationships between productivity, quality, and measurement, these classes of measures are identified, and "dominant measures" are grouped according to the maturity levels defined by the Software Engineering Institute's Capability Maturity Model for Software. Finally, it is noted that comprehensive planning for the cultural changes that must accompany I/S measurement is critical to the successful deployment of the process.

TEXT: The phrases "market-driven quality," "total customer satisfaction." and "total **quality management** (TQM)" often come up in executive conversations today because of the growing awareness that the ability of a business to compete effectively depends on increasing quality and productivity.(1) Many companies are actively reengineering their business processes to achieve quality and productivity improvements. Information systems (I/S) technology often plays an important role in business process re-engineering initiatives. As a result, I/S customers throughout the enterprise are demanding higher-quality software applications that do more and are delivered faster. Software managers, like their line-of-business customers, are struggling to improve quality and productivity. Unfortunately, higher quality and greater productivity are often viewed as competing, rather than complementary, goals. Consider, for example, organizations that seek to improve application development quality through increased testing. These organizations usually find that quality improves, but overall development productivity suffers. The additional testing requires additional time and expense, both for the testing effort and for correcting the defects that testing uncovers.

The conclusion that quality improvements come at the expense of productivity is caused by not viewing the relationship between quality and productivity as part of a total system. By analyzing the software development life cycle as a whole, I/S organizations can look at ways to build quality into the whole system, from initial conceptualization of a software application, all the way through maintenance and obsolescence.

This paper is directed at I/S organizations that are considering the implementation of a software measurement process as a means to quantify and improve their value to the enterprises they serve. We hope that these organizations will benefit from an overview of measurement principles, a translation of these principles into I/S terms, an appreciation for the hierarchical nature of meaningful measures, and an awareness of the cultural and organizational aspects of deploying a measurement process.

Quality drives productivity. Many experts in quality, including W. Edwards Deming, Joseph Juran, and Philip Crosby, have shown that the key to improving product quality lies in improving the quality of the process by which the product is made. Reduced rework and improved productivity are direct results of improving the quality of the production process. This conclusion is supported by a Federal Quality Institute finding in the

mid-1980s that "productivity is a by-product, a result of quality improvement."(2) Software organizations are more frequently reporting that increased quality results in increased development productivity.(3)

If the concept that quality drives productivity seems obvious, why is it that many I/S TQM efforts are not underway? One reason may be that I/S has been traditionally viewed by an enterprise as an expense center. Because expense centers do not interface directly with external customers and do not usually produce products used by these customers, they are typically the last areas targeted by TQM initiatives.

A more fundamental reason, in our opinion, is that effective quality initiatives require clear definitions and explicit measures of quality. Many I/S organizations have failed to define and measure quality in their customers' terms. This failure is a failure to communicate. High costs are associated with the communication failure. Having failed previously to work with end users to define application quality in customer terms, I/S is often not included in the early stages of the line-of-business system planning activities of an enterprise and misses the opportunity for effective partnership with them. As a result, I/S often does not provide to the enterprise the full value of which it is capable.

In working with customers to define software quality, I/S has to recognize that quality includes both utility (usefulness) and usability. Utility refers to how much the application helps the user to complete his or her work. Usability is a combination of attributes such as user-friendliness and reliability. In order to be successful in meeting customer expectations, I/S development organizations must work with their customers to define quality for application development, and then must develop metrics that will help them manage to that definition of quality.

Ultimately, any definition of quality has to address both the products that I/S delivers and the software processes it uses to deliver them. These two broad dimensions of quality are often represented by the following questions:

1. Are we doing the right things?

2. Are we doing things the right way?

Table 1 illustrates the two dimensions. (Table 1 omitted)

Measurement drives quality. Quality experts are certain that measurement is essential to improving quality. According to Dr. Curtis Reimann, 1989 Chairman of the Board of Overseers of the Malcolm Baldrige National Quality Award, "The number one factor common to companies scoring high in quality was that they were quantitative and had instituted measurement processes." In other words, measurement drives quality. Well-designed measurement focuses on goals; reciprocally, well-defined quality is measurable.

Measurement is an essential component of total **quality management** . The old axioms are true: You cannot manage what you cannot measure; what is not tracked is not done.

Well-implemented measurement is an ongoing process of measuring work processes and products, finding out where the organization is compared to where it needs to be, and analyzing data to identify opportunities for improvement.(4) Measurement information is essentially management information. Well-designed measures identify the current capabilities of I/S, highlight opportunities for process improvement, facilitate goal setting, mark progress toward goal attainment, and enable benchmark comparisons with other organizations.

When used for process improvement, measurement can actually improve job satisfaction and morale because process improvements help people to work more effectively. The most successful measurement initiatives place a strong emphasis on employee involvement. Effective measures are not

measurements of the people. They are measures of work processes made by the people and for the people so that process improvements can be defined and implemented. Effective measurement results in higher-quality products and in enhanced pride that people take in delivering these products to their customers.

The best metrics practices are linked to quality improvement efforts that harness the efforts and intelligence of everyone in the organization to find ways to do things better, from start to finish. The end result is lowered costs and increased productivity.

In application development, the cost of discovering and eliminating a defect increases dramatically as the application proceeds through the development life cycle. By moving defect detection efforts up to the early phases of application development, defect detection can be 33 times more cost-effective than testing done at the end of development.(5)

Hierarchy of measures. One of the most common problems that quality and metrics consultants encounter in I/S organizations is the absence of the concept of hierarchical information needs. Consider the case of the CIO (chief information officer) at a very large company, responsible for an I/S organization of 5000 people spread throughout several corporate divisions. The enterprise had already adopted TQM as its approach to improving its bottom line: market share and productivity. He knew appropriate measures were essential, so, being a business manager who had not come through the ranks of I/S, he asked I/S to propose the executive-level measures.

The I/S organization recommended that the executive-level measures be lines of code and person hours. These measures were ultimately rejected by the CIO because there was no way for him to relate such basic, low-level measures to the business objectives at his level. The measures told him nothing about how smoothly I/S development processes worked or whether the processes were under control, and they told him nothing about how satisfied S customers were, or how much I/S contributed to the performance of the customer organizations. What the CIO needed were the results of translating information up through the hierarchy of information needs of the organization.

Effective measures serve as a framework for defining shared goals and for communicating how the goals of each unit directly support organizational goals. Measures empower people at all levels by providing the data needed to make fact-based decisions. These data provide objective information about the working of the processes of the organization. Acting effectively upon the information requires management to shift the focus from personnel performance to process performance and from individual behavior to team behavior.

Successful approaches to establishing and using measurement recognize that each level of the organizational hierarchy has different information needs. The information needs will tend to be hierarchically structured: each level of the organization has to roll up information from its level into reports for the the next higher level. When measurement is designed to meet the hierarchy of information needs, the measures and metrics used tend to create a hierarchy of measures.

A hypothetical example may help to explain the concept of measurement hierarchies. Let us assume that the goal of one CIO is to increase responsiveness to the users of I/S by delivering applications faster. To do that, the organization has to shift more effort to development, away from maintenance. To support the CIO's goal of improved responsiveness, the I/S director establishes two goals: (1) increasing development quality and (2) decreasing maintenance costs. By achieving these goals, he or she will then be able to direct more of the total I/S effort to developing new applications.

For the application development managers, this goal translates to improving the quality of new applications delivered to users (I/s customers) and

improving the maintainability of applications that are turned over to the maintenance group (the internal "customer" of the new-development group). Both new-development and maintenance project managers agree that reusing code is a good way to reduce error rates in new applications, thereby lowering the cost of delivering the application. They also agree that maintenance efforts should similarly benefit. Thus, they agree that quality and maintainability increases will initially be tracked by measures of code reuse and rework effort. The project managers decide to measure reuse and rework as follows:

* Percent code reused = Reused lines of code / Total lines of code

* Percent reusable code developed = Reusable lines of code developed / Total lines of code developed

* Percent rework = Total person hours devoted to rework / Total person hours of effort

Because each organization has its own particular culture, goals, and values, each organization needs to establish and internalize its own measures. This set of measures should be constructed to serve as critical business indicators, showing at a glance how the organization is doing.

DIMENSIONS OF I/S MEASUREMENT

An essential step toward effective measurement is deciding what to measure, how to measure it, and how to use the measurement data. As discussed earlier, there are two broad dimensions of I/S quality: whether r/s is doing the right things, and whether it is doing them in the right way. Both of these dimensions need to be measured. This section presents two classes of I/S measures to be considered: business value measures and I/S process and product measures. Both classes of measures are needed to answer the question: Are we doing the right things in the right way? I/S process and product measures address the right way; business value measures address the right things.

Doing things right--I/S process and product measures. Doing things right means producing quality products and services as efficiently (which also means as cost-effectively) as possible.

I/S process and product measures provide insights into the capability of I/S to deliver quality applications. Process and product measures gauge process efficiency and outcome predictability by tracking such things as differences between estimates and actuals (e.g., effort, size, cost, maintenance requirements, and customer satisfaction). Information about project-specific attributes (e.g., project development platform and development methodology) is also required to appropriately group projects together when performing quantitative analysis.

Process and product measures are required to identify opportunities for meaningful improvement and to identify results (e.g., phase and cycle time, defects, rework) that significantly differ from the norm. The factors that contribute to favorable results can be propagated to other projects. Factors that contribute to undesirable results can be eliminated, thereby improving the software development and delivery processes.

Process and product measures are used to answer questions like "Are things getting better? Why? Are our improvement efforts effective?" Comparisons between pilot project results and historical norms, as well as overall trends, can test and demonstrate the effects of process changes.

The application developer's and tester's perspectives. Involving developers and testers in the measurement definition process will, in the long run, result in better estimates and better teamwork. Developers want measurable descriptions of the product to be developed, including customer expectations and quality requirements, in addition to project schedules and milestones. The developer is interested in measures that demonstrate (and support estimates of) how much effort is required to reach the target

milestones and to deliver an error-free product. Providing straightforward ways to track current effort enables developers to be self-monitoring and self-correcting. When developers are self-correcting and when testers can see a record of the developers' testing activities, system testing costs will likely decrease. The tester wants to know how well the development group has inspected for errors, how many errors have been captured and corrected, and how large and complex the software is. Testers hope that quality will be built in, because they know how costly it is to try to "test it in."

The project manager's perspective. The project manager is expected to keep the project on plan. The project manager's measures need to track the project closely enough to provide enough lead time to anticipate problems and make timely course corrections. The project manager also needs to make and revise the project plan based on new information or changing conditions. Project estimates and revisions require accurate historical data in order to make reasonable budget and schedule projections.

Common project management measurement requirements include regular reports of progress (e.g., task starts and completions, milestone attainments), requirement changes that impact cost and scope (e.g., estimated versus actual impact of change requests), and defects (e.g., number of defects detected and defect severity broken out by phase).

The I/S development director's perspective. The I/S development director's perspective spans multiple projects. The development director needs measures that highlight meaningful differences between projects. The objective of the I/S development director's measures is to provide information for optimizing the software development processes of the organization. As owner of the development function, the I/S development director needs measures to identify factors contributing to or inhibiting quality and productivity, measures to evaluate the impacts of new tools and techniques, and measures to provide facts upon which resource allocation decisions can be made.

Business value measures--doing the right things. i/S products and services derive business value from improving the business performance of the enterprise by working with the lines of business to document, understand, and streamline their processes. Business value, then, is derived from measured enhancement in business performance. The amount of change can be measured by comparing initial performance to performance enhanced by I/S.

Business performance is defined differently across different enterprises, depending on values specific to their defined missions. In order to maximize its business performance enhancement potential, I/S needs to be aware of, and plan according to, the values or strategic priorities of the planners of the enterprise. Table 2 presents commonly recognized categories of business value.(6) (Table 2 omitted) Each enterprise will place a different emphasis on different categories. Within each enterprise, each line of business may similarly emphasize categories differently, depending on current and future business needs.

The CIO's perspective. The CIO is in a uniquely advantageous position, bridging senior management planning activities that define the strategic directions of the enterprise and the automation and process re-engineering expertise of I/S. The stronger the bridge between enterprise planning and I/S planning, the more likely it is that S can maximize its contribution to the enterprise. Linking the two levels of planning facilitates the effective benefits assessment of proposed I/S efforts and promotes effective prioritization of I/S projects.

Thus, the CIO looks at the contributions that I/S can make to the enterprise as a whole and at the contributions it is currently making. When evaluating the contributions made by Us application development efforts, the CIO requires timely, meaningful measures of customer satisfaction, I/S internal quality and productivity, and I/S business value contributions

The CIO is also interested in external benchmarking to answer questions like:

* How do we stack up against other people? Against the best?

* If we are not the best, what can we learn from others who are?

* How does our experience with new technologies and methodologies compare with that of others?

Measures of business value provide the CIO with the data needed to understand the big picture: to know how much value I/S applications bring to its customer organizations, and to determine how much the value of I/S could increase through improvements in the capability of I/S to deliver quality applications. On the basis of current and potential business value, the CIO can make fact-based decisions when allocating the resources required for application development, application customization, maintenance, and process improvement.

Measuring I/S quality and value. The preceding subsections have looked at various factors that can be measured to assess whether I/S application development efforts are effective (focused on the right things) and whether application development is efficient (being done the right way). This subsection suggests a way to group I/S measures into measures of effectiveness and efficiency.

I/S effectiveness means focusing effort and resources where they will have the greatest positive business impact. Measurements of effectiveness usually include external perspectives:

* Customer satisfaction-Customer satisfaction can be measured by surveying user satisfaction with products and services supplied by I/S. Techniques to measure customer satisfaction may include customer surveys, evaluation questionnaires following up each delivery of a product or service, and customer forums, focus groups, or user groups. Customer surveys should include the business process owners, line-of-business executives, and application end users.

* Business impact-Business impact can be measured by evaluating I/S products and services in terms of their contributions to the competitiveness, productivity, and flexibility of the enterprise in daily business; that is, how much do I/S products enhance business performance? Assessing business impact includes follow-up on the "business cases" supplied by business units when requesting new I/S products or services, as well as pre-and post-install surveys. Business impact measures may include the business value of new markets or new revenue streams that an I/S product has enabled.

* Strategic alignment--Strategic alignment can be measured by evaluating the deployment of I/S products and services vis-a-vis the strategic goals of the enterprise. The alignment value of I/S is measured in relation to the alignment of I/S with business priorities. Strategic alignment may measure I/S effort expended per business unit in relation to the strategic value of that unit to the business. Alignment measures may also consider how closely I/S planning and enterprise planning activities are integrated.

Efficiency is the ratio of effective work to the energy expended in producing it. Efficiency measures the ability to produce a desired effect with a minimum of effort, expense, or waste. I/S measurements of efficiency are inward-focused; they relate to I/S internal activities and are typically technically oriented. Because they measure product against work effort, they consider:

* Portfolio analysis-Portfolio measures may include the total number of applications, total size of the portfolio, portfolio cost per function point across lines of business, and annual trends in product quality or delivery cycle time. The measures should also include services offered,

such as consulting, help desk, product documentation, and product training.

* Process analysis-Process analysis measures evaluate the repeatability and predictability of a process. They show whether or not a process is under control. The key here is how dependable the process is: Can budgets and schedules be estimated accurately? Can product quality be reliably predicted? Process analysis measures examine the rate at which work flows through each step of the process: the rate at which user requirements are translated into function delivered to the user. What percentage of overall development time is spent on a given phase of the development cycle, such as defining and validating requirements? How many errors are detected and corrected during each phase? How well are requirements traced through the process to delivery of the application?

* Cycle time-Cycle time measures responsiveness and shows the rate at which user requests for new products, enhancements, and services are fulfilled. Cycle time measures may include variances between projected and actual completion times, average and median times to deliver new applications and enhancements, and the rate at which backlog service requests are moved into active project status (a kind of "inventory turn" measure).

Figure 1 summarizes the factors that contribute to I/S quality and value. (Figure 1 omitted)

ALIGNING PROCESS ANALYSIS MEASURES WITH MATURITY

In the earliest years of software development, and in many organizations today, the goal of development was and is just to get the product out. Thirty years ago, there was no science of software engineering. People invented ways to build software at the same time as the software products were being built. Processes were ad hoc, and success often depended upon the "guru" leading the project. Fortunately, much has been learned over the past few decades.

A considerable body of knowledge has evolved, improving the ways in which software is designed, developed, and delivered. A new science, software engineering, has been born. Software engineering practices bring rigor and discipline to software organizations. Studies of commercial and governmental software organizations by the Software Engineering Institute (SEI) at Carnegie Mellon University have resulted in the definition of a software process Capability Maturity Model (CMM).(7) The process Capability Maturity Model distinguishes five levels of increasing maturity and capability.

Although the CMM does not provide an explicit formula for improving individual development organizations, it relies on empirical data that show a strong link between the maturity of the processes used within the software organizations and their ability to produce predictable results. Interestingly enough, the thesis that the ability of an organization to deliver high-quality products and services is linked to process maturity is also embodied in the criteria for the Malcolm Baldrige National Quality Award. The quality capability model of the Baldrige assessment also uses five levels of increasing capability.

SEI's model was originally developed by Watts Humphrey, who described his approach to defining a framework for software process improvement as "roughly parallel" to a total **quality management** model mapped out by Philip Crosby.(8,9) Thus, it is not accidental that there is some correspondence between quality models and the Capability Maturity Model. Each suggests that the best way to improve software development quality and productivity is to improve the software development process. The five levels of the Baldrige Score Range, Crosby's **Quality Management** Maturity Grid, and the SEI Capability Maturity Model are summarized in Table 3. (Table 3 omitted) Crosby's Maturity Grid also shows the percentage of sales the cost of poor quality (COPQ) represents at each level of maturity.

Capability Maturity Model. Table 3 also illustrates the basic premise of the CMM: as software organizations mature, the processes they use mature. The CMM identifies five levels of software engineering process maturity ranging from Level 1, "initial" (the relatively chaotic situation that is still most common today), to Level 5, "optimized," the level of greatest maturity and highest quality.

At Level 1, the goal is simply to deliver software. There is no predictability for completion time, cost, quality, or functionality. Development processes are largely ad hoc. Usually, the only measures that are reliable are post-ship measures taken after the software product I/S delivered to the user. These measures often bring unpleasant surprises to management.

At Level 2 of software development maturity, the goal is project control. This goal is usually translated into schedule control. At this stage, projects are schedule-driven; cost control is often difficult because of the need to throw extra resources (such as large amounts of overtime) into the project so that a product is ready on time.(10) Unfortunately, often not enough hard historical data are available to enable reasonable schedule projections, so time begins to run out. In this "process," it is not unusual for functionality and quality requirements to fall by the wayside. As a result, costs to achieve customer satisfaction after the product is released can be very high, especially if there is a large installed base of users.

Level 3 works on controlling the product by assuring that all product requirements are met and quality-controlled. At Level 3, the expected product is produced on or close to schedule, but costs are still not under control. This situation is often attributable to the increased commitment to defect detection and removal. Because detection and removal efforts do not yet improve the process that introduces the defects, the cost of attaining quality is high.

At Level 4, the goal is to control the process by which the product is produced. Only by controlling the process can the costs of products, projects, and customer satisfaction be controlled. The expected product is delivered when predicted at the predicted cost. Now the organization can begin to improve processes and, by doing so, reduce cycle time, reduce costs, and produce better products.
At Level 5, the software organization builds on the foundation of control and improvement to establish a culture of continuous process improvement. Software development efforts focus on optimizing both quality and productivity. Although it is not a focus of the CMM, we can assume that the software organization itself has probably achieved much tighter integration with the business as a whole. As a result, measures that track the value to the business of the I/S organization as a whole are probably relatively mature.

Capability Maturity Model dominant measures. Table 4 shows samples of the measures currently advocated by the SEI (dominant measures) at each level of maturity.(11) (Table 4 omitted) Each change in focus for each level of maturity dictates a change in the measures used to manage software development. As the table illustrates, the dominant measures shift as the level of process maturity increases. This shift reflects three common axioms of measuring:

* What is not tracked is not done.

* You cannot manage what you cannot measure.

* What you do measure is what you get.
Let us look at those statements more closely. The information that measurement supplies is essentially a management information system (i.e., a system that supports decision-making). For that reason, the dominant measures relate to what the organization is trying to manage. As I/S organizations reach higher levels of software engineering maturity, the

management information needs become more sophisticated. Processes are better designed, understood, and followed. Opportunities for "fine tuning the development engine" increase, driving the need for increased precision from existing measures and for collecting new measures.

The relationship between the I/S level of maturity and information needs is reflected in the table by changes in the dominant measures. As each succeeding level of maturity is reached, and additional measures are put in place, the dominant measures from the preceding level will occupy a smaller share of management interest and attention than it did previously.

The various sets of measures and metrics that the organization uses can be portrayed as gauges on an instrument panel. In arranging the instrument panel of the organization, managers might typically position the dominant measures as "continuous readout" dials (like speedometers and odometers). Other instrument panel areas would be assigned to "warning indicators" for measures and metrics that are of interest to management only when immediate action is required (like low oil lights and door-ajar bells). As the organization reaches higher levels of maturity, new dominant measures will reconfigure the instrument panel: The new dominant measures will become continuous readout gauges and the dominant measures of the prior level will become warning indicators. Determining the appropriate S measures is therefore an ongoing process. Attaining a new level of software engineering process maturity or recognizing a shift in the enterprise business value model are indicators that it may be time to repeat the measurement selection process. Each time the measurement selection process is repeated, the effort required to define, collect, and analyze the measures should decrease.

CULTURAL AND ORGANIZATIONAL DIMENSIONS OF MEASUREMENT

Successfully introducing measurement into an organization almost always requires significant cultural changes. Comprehensive planning for the cultural changes that must accompany I/S measurement is critical to the successful deployment of the process. Most people resist change. By understanding the dynamics of change, and carefully planning its introduction, I/S management can minimize stress for everyone involved. In fact, organizational change can be a positive, team-building experience. Successfully managing change requires coordination, communication, and commitment. Management must repeatedly reaffirm their commitment to the ultimate benefits gained from the change. These efforts are needed to reduce fear and stress. A strong communication effort should precede and constantly accompany the formal implementation of the measurement in order to minimize or forestall negative reactions people may have to the change.

Organizational change is most successful when the entire organization:

* Recognizes the need to change

* Establishes and communicates the new shared vision

* Endorses the plan to arrive there

* Communicates, celebrates, and rewards progress

Cultural success factors for measurement. Optimizing the capacity of the organization for change is I/S management's challenge. Planning for successful change must address the culture of the organization as well as its structure. Appropriate leadership and sponsorship of the change are extremely important. Figure 2 illustrates the primary contributors to successful organizational change. (Figure 2 omitted)
Carefully selected measures. The good news from measurement is: What you measure is what you get. The bad news from measurement is also: What you measure is what you get.

Regardless of how measures are used, the act of gathering measures focuses attention on what is being measured. People will naturally try to produce

what is being measured, whether it be lines of code, fewer errors, or ease of use. Careful thought should go into selecting the measures that are right for a particular set of I/S information needs, objectives, and culture. For example, productivity targets stated only in terms of lines of code may cause developers to increase the number of lines of code (without adding any value to the application) and hence "improve" productivity as defined by the measurement plan.

Visible progress. The Quality Assurance Institute suggests that the plan should include milestones that mark short-term successes. Celebrating the attainment of these milestones reinforces the momentum behind the plan. In addition, focusing attention on positive results firm the measurement process serves to affirm the commitment of management which, in turn, reinforces acceptance of the measurement process by the organization.

Readiness for change. Healthy organizational change begins when the organization, as a whole, recognizes the need to change. Many experts believe that three phases are involved: unfreezing, moving, and refreezing. Unfreezing is the process of recognizing the existence of a problem. Surveys or assessments (either self-assessments or external assessments) can assist the unfreezing process. Moving involves establishing a vision of the "brave new world," or the scenario after change takes place, as well as a plan to get there. Refreezing involves the implementation of the plan, monitoring progress, and making course corrections.

Consistent goals. The establishment of consistent goals from top business management to top I/S management to project management to the programmer is vital to the success of the program.

Top management goals such as return on investment or increased market share must be decomposed to specific measures for the project manager and the programmer. How the measures are combined to evaluate progress toward organizational goals should be clearly articulated and endorsed. In this way everyone can see how individual efforts relate to the big picture. For example, improved market share depends on improved quality, which depends on fewer defects, which depends on early defect identification, which depends on process improvement.
The real test of a measurement process is the degree to which everyone can make the translation from top management goals to the goals that each person is being asked to achieve. By passing this test, the measurement process helps the individual identify with the enterprise objectives and feel a part of them.

Organization and sponsorship. The owner of each process being measured must sponsor the deployment of its measurement. A team approach is highly recommended to determine the appropriate measures for a given process. Key people who are participants in the process being measured should be part of the planning and implementation team. Participants may also include measurement subject matter experts from quality assurance or the **software engineering process group** ,(12) but in an advisory capacity. The key people on the deployment team should be the "informal" leaders in the organizations they represent. Their buy-in, or commitment to participate, is essential if the deployment is to succeed.

Management behavior. The managements of organizations that have successfully undergone organizational change have several traits in common. Among these are that management:

* View change as a process and prepare their organizations for evolution
* Understand that resistance is to be expected and develop plans to manage and minimize potential side effects of resistance focus on team building and provide teamwork skill training and incentives for teamwork

* Create an environment where change comes from the "grass roots" but is directed toward a common vision

Assessing readiness for change. Change management experts have identified

the following factors when assessing organizational readiness for change: (13)

History of change--the prior experience of the organization in accepting change. Has a measurement process been attempted before? What happened?

* Clarity of expectations-the degree to which the expected results of change are shared across various levels of the organization. The classic example of conflicting expectations is the case in which a vague "quality" emphasis program is announced, perhaps even with a designated "quality executive," but the day-to-day management focus remains clearly fixed on meeting schedules, rather than on improving quality.

* Origin of the idea or problem--the degree to which those most affected by the change initiated the idea or problem that the change solves. Packaged, "quick fix" measurement solutions often encounter "not invented here" problems. Similarly, measurement' promoted by a department, external to the people involved in the process being measured, may not be well accepted. For example, developers frequently (and rightfully) resent interference from "watchdog" organizations that dictate what should be measured and why. This lack of acceptance is especially true when the people being asked to change do not believe there is a need to change.

* Support of top management--the degree to which top management sponsors the change. Top management support and involvement are evident when measures of the progress of the entire organization toward its goals are shared throughout the organization, when management requires that measurement results be included in all status reports, and when management authorizes allocation of sufficient time in the work week for people to produce, collect, analyze, and use measures.

* If top management pays only lip service to the change and does not provide leadership for change, if management fails to monitor the adoption of change and does not allocate the resources necessary to bring about the change, the rest of the organization will not take the change seriously. The change effort will sputter and die. Future cultural change efforts will face an even greater challenge.

* Compatibility with organizational goals--the degree to which the proposed change corresponds to past and present organizational practices and plans. For example, if management claims that "quality" is an important goal but only uses schedule-driven measurements, the organization will recognize the dichotomy and thwart the measurement process.

CONCLUDING REMARKS

What to measure depends upon management information needs which, in turn, relate to enterprise-wide priorities. Senior I/S management wants to know if the I/S organization is doing the right things, in the right way. Both business value ("right thing") measures and I/S process and product ("right way") measures are required to satisfy the information needs of I/S management. The I/S measures should be organized hierarchically because of the hierarchical and interlocking nature of organizational, departmental, and project goals and information needs. Each level of the I/S organization must satisfy its own information needs and the information needs of the next higher level. The degree to which these information needs are satisfied is directly related to the degree to which the information needs can be quantifiably translated into a hierarchy of measures.

Although a picture of the full hierarchy may help communicate the ultimate vision of the measurement process, few organizations are capable of implementing the grand vision all at once. The ability to achieve an optimal hierarchy of measures and metrics ranging from application development projects up to and beyond the CIO depends on the following factors:

1. Complete understanding of the ability of the application development organization to measure (14)

2. Full alignment of business and quality goals up and down the organizational hierarchy

3. Selection and use of the most appropriate measures and metrics to track achievement of the aligned goals

4. Thorough assessment of the cultural changes required to integrate measurement into the way in which the organization carries out its business

5. Comprehensive planning for metrics implementation

The full value of a measurement process can be realized only if all of these factors are present and properly balanced.

ACKNOWLEDGMENTS

The authors would like to acknowledge the following people for their patient review of and many helpful suggestions for this paper: Phil Braverman, Jim Constans, Jim Freeman, Maryanne Kuzniar, Bob McNeil, Steve Noble, Dennis Orton, Carolyn Schorr, Steve Thomas, and Barry Young.

CITED REFERENCES AND NOTES

1. Critical Issues of Information Systems Management for 1991, The Fourth Annual Survey of I/S Management Issues, Index Group (December 1990) and critical Issues of Information Systems Management for 1992, The Fifth Annual Survey of I/S Management Issues, Index Group, Cambridge, MA (December 1991).

2. L. Dobyns and C. Crawford-Mason, Quality or Else, Houghton MiWin, Boston (1991), p. 168.

3. Conference Proceedings, quality Week 1991, Software Research Inc., San Francisco (May 14-17, 1991).

4. This process is embodied in W. Edwards Deming's "Plan-Do-Check-Act" cycle and in Juran's continuous improvement model and is integral to H. James Harrington's business process improvement model. Inspection in Ultra

5. G. W. Russell, "Experience with Large Scale Development," IEEE Software s, No. 1, 25-31 (January 1991).

6. M. M. Parker and R. J. Benson with H. E. Trainor, Information Economics, Linking Business Performance to Information Technology, Prentice-Hall, Inc., Englewood Cliffs, NJ (1988).

7. M. C. Paulk, B. Curtis, M. B. Chriss is, et al., Capability Maturity Model for Software, Version I.I, CMU/SEI-93TR-24, Software Engineering Institute, Pittsburgh (January 1993).

8. W. S. Humphrey, Managing the Software Process, Addison-Wesley Publishing Co., Reading, MA (1989), p. 4.

9. P. Crosby, Quality Is Free, McGraw-Hill Book Company, New York (1979).

10. A schedule/cost/quality model developed by Quantitative Software Management, Inc., suggests that shortening the schedule by adding project team members may adversely affect quality.

11. J. H. Baumert and M. S. McWhinney, Software Measures and the Capabilig Maturity Model, CMU/SEI-92-TR-25, Software Engineering Institute, Pittsburgh (September 1992).

12. W. S. Humphrey, op. cit., p. 31.

13. M. M. Danziel and S. C. Schoonover, Changing Ways: A Practical Tool for Implementing Change Within Organizations, AMACOM, a Division of American Management Association, New York (1988), pp. 15-16.

14. This paper suggests that the ability of the organization to measure is linked to its level of software engineering process maturity and that its "dominant measures" should be selected based upon the level of maturity of the organization.

GENERAL REFERENCES

The following selections are suggested for those interested in learning more about software measurement, quality, and productivity.

SOFTWARE MEASUREMENT

A. J. Albrecht and J. E. Gaffney, "Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation," IEEE Transactions on Software Engineering SE-9, No. 6, 39-64 (November 1983).

B. W. Boehm, Software Engineering Economics, Prentice-Hall, Inc., Englewood Cliffs, NJ (1981).

R. C. Camp, Benchmarking-The Search for Industry Best Practices That Lead to Superior Performance, ASQC Press, Milwaukee (1989).

C.-K. Cho, quality Programming: Developing and Testing Software with Statistical Quality Control, John Wiley & Sons, Inc., New York (1987).

S. D. Conte, H. E. Dunsmore, and V. Y. Shen, Software Engineering Metrics and Models, Benjamin/Cummings Publishing Co., Menlo Park, CA (1986).

B. J. Dreger, Function Point Analysis, Prentice-Hall, Inc., Englewood Cliffs, NJ (1982).

A. S. Duncan, "Software Development Productivity Tools and Metrics," Proceedings of the 10th International Conference on Software Engineering, Singapore (April 11-15, 1988), pp. 4148.

R. B. Grady, Practical Software Metrics for Project Management and Process Improvement, Prentice-Hall, Inc., Enplewood Cliffs, NJ (1992).

R. B. Grady and D. L. Caswell, Software Metric's: Establishing n Company-Wide Program, Prentice-Hall, Inc., Englewood Cliffs, NJ (1987).

W. S. Humphrey, Managing the software Process, Addison Wesley Publishing Co., Reading, MA (1989).

IEEE Software Productivity Measurement Committee, Dmfi Standard, IEEE Computer Society (1990).

C. Jones, Applied Software Measurement, McGraw-Hill Book Co., Inc., New York (1991).

T. McCabe, "A Complexity Measure," IEEE Transactions on Software Engineering SE-2, No. 4, 308-320 (December 1976).

H. D. Mills and J. H. Moore, "Bringing Software under Statistical Quality Control," quality Progress XXI, No. 11, 52-55, ASQC Press, Milwaukee, WI (November 1988).

J. Musa, A. Iannino, and K. Okumoto, Software Reliability: Measurement, Prediction, Application, McGraw-Hill Book Co., Inc., New York (1987).

H. M. Parsons, "What Happened at Hawthorne?", Science, No. 183, 922-932 (March 8, 1974).

L F Urwick and E. F. L. Brech, The Hawthorne Investigations Volume III: The Making of scientific Management, Management Publications Trust, London (1949).

R. Werling, Data Collection System for Estimating Development Cost, U.S. Air Force Contract No. F33615-85-C-5123, Defense Technical Information Center, Alexandria, VA (30 SEP 86).

H.-T. Yeh, Software Process quality, McGraw-Hill Book Company, New York (1993).

H. Zuse, Software Complexity: Measures and Methods, Walter de Gruyter, Inc., Hawthorne, NY (1990).

QUALITY
R. Aguayo, Dr. Deming, Simon & Schuster, New York (1990).

K. R. Bhote, "Motorola's Long March to the Malcolm Baldrige National Quality Award," National Productivity Review S, No. 4, 365-375 (Autumn 1999).

C. W. Burrill and L. W. Ellsworth, quality Data Processing Profit Potential for the 1980s, Burrill-Ellsworth Associates, Inc., Tenafly, NJ (1982).

W. E. Deming, Out of the Crisis, Massachusetts Institute of Technology, Center for Advanced Engineering, Cambridge, MA (1986).

W E. Deming, quality, Productivity and Competitive Position, Massachusetts Institute of Technology, Center for Advanced Engineering, Cambridge, MA (1982).

J. F. Halpin, Zero Defects-A New Dimension in Quality Assurance, McGraw-Hill Book Co., Inc., New York (1987).

H. J. Harrington, Business Process Improvement-The Breakthrough Strategy for Total Quality, Productivity, and Competitiveness, ASQC Quality Press, Milwaukee (1991).
H. J. Harrington, Excellence-The IBM Way, ASQC Quality Press, Milwaukee (1988).

M. Imai, Kaen-The Key to Japan 's Competitive Success, Random House, New York (1986).

C. L. Jones, "A Process-Integrated Approach to Defect Prevention," IBM Systems Journal 24, No. 2, 150-167 (1985).

G. G. Schulmeyer, Zero-Defect Software, McGraw-Hill Book Co., Inc., New York (1990).

Software Design Inspections, Software Productivity Research, Burlington, MA (1990).

User Satisfaction Survey questionnaire, Software Productivity Research, Burlington, MA (1989).

1990 Application Guidelines: Malcolm Baldrige National quality Award, U.S. Department of Commerce, National Institute of Science and Technology, Gaithersburg, MD (1990).
PRODUCTIVITY

T. K. Abdel-Hamid, "Investigating the Cost/Schedule tradeoff in Software Development," IEEE Software 7, No. 1, 97105 (January 1990).

B. W. Boehm, "Improving Software Productivity," Computer 20, No. 9. 43-57 (September 1987).

B. W. Boehm, "A Spiral Method of Software Development and Enhancement," Computer 21, No. 5, 61-72 (September 1988).

F. Brooks, "No Silver Bullet: Essence and Accidents of Software Engineering," Computer 20, No. 4, 10-19 (April 1987).

R. Carlyle, "Leaping Ahead in Software Productivity," Datamation 35, No. 23, 2232 (December 1, 1989).

W. B. Chew, "No-Nonsense Guide to Measuring Productivity," Harvard Business Review 66, No. 1, 110-118 (January February 1988).

L. Cohen, "Quality Function Deployment: An Application Perspective from Digital Equipment Corporation," National Productivity Review, 197-207 (Summer 1988).

T. DeMarco and T. Lister, Peopleware: Productive Projects and Teams, Dorset House Publishing, New York (1987).

C. Jones, Programming Productivity, McGraw-Hill Book Co., Inc., New York (1986).

B. King, Better Designs in Half-the Time: Implementing quality Function Deployment in America, GOAL/QPC, Methuan, MA (1987).

J. Martin, "Gain Management Support by Measuring Productivity," PC Week (January 8. 1990), p. 76.

J. Martin, "Integrated CASE Tools a Must for High Speed Development," PC Week (January 22, 1990), p. 78.

R. J. Norman and J. F. Nunamaker. Jr.. "Case Productivity Perceptions of Software Engineering Professionals," Communications of the ACM 32. No. 9, 11021108 (September 1989).

Cherlene ("Chuck") Walrad 12 Boke Circle, Mill Valley, California 94941-4604. Ms. Walrad is a software management consultant specializing in cross-life-cycle quality and productivity and their measurement. She has instituted programs and techniques to introduce total **quality management** within a variety of commercial software companies and I/S organizations. She has firsthand knowledge and experience of the corporate-wide quality programs at Xerox, Intel, and IBM. She has implemented a wide range of software quality and productivity processes, spanning all phases of the software life cycle and promoting cross-functional teamwork. These processes include design and development inspections and reviews, phase gating, formal configuration management, test automation, measurement and tracking programs to identify and eliminate sources of customer dissatisfaction and to reduce support costs, and quality audits of third-party software vendors. Prior to becoming a consultant in 1987, Ms. Walrad managed the design, production, or support, or both of the latter, of more than two dozen commercial and government software products, including mission-critical systems for U.S. Intelligence and the Commission of the European Communities. She has had budget and schedule responsibilities for multimillion-dollar software projects as well as small projects, has managed small teams as well as large organizations, and has managed remote operations in Europe, Mexico, and Japan. Ms. Walrad has been an invited speaker at conferences, at the U.S. Department of Defense, the Department of Commerce, the Central Intelligence Agency, and the National Academy of Sciences. She has taught at the university level, co-authored a college textbook, been selected for membership in Mensa, and is included in Who's Who in the West and Who's Who of Global Business Leaders.

Eric Moss IBM Consulting Group, 1111 Broadway, 17th Floor, Oakland, California 94607. Mr. Moss is part of the IBM Consulting Group and is a member of a team developing IBM's consulting methodology for application development quality, productivity, and measurement services. He has consulted with several IBM internal I/S organizations to implement function point analysis. Prior to joining IBM, he developed integrated financial modeling systems for several major public utilities, authored a planning policies and procedures manual for a major West Coast utility, and developed statistical sales forecasting models for a major consumer package goods company. Mr. Moss holds a Bachelor of Science degree in management science from Carnegie Mellon University and a Master of Business Administration from the University of California, Los Angeles. He has also been recognized as a Certified Quality Analyst by the Quality Assurance

Institute.

DESCRIPTORS: Systems development; Information systems; Quality control; Productivity; Roles; Measurement
CLASSIFICATION CODES: 5240 (CN=Software & systems); 5320 (CN=Quality control)